PDM Workbench

PDM Workbench Release 3.6 for Aras Innovator

Installation & Administration Manual

Copyright

© 2005-2014 T-Systems International GmbH.

All rights reserved. Printed in Germany.

Contact

T-Systems International GmbH
GDC Product Lifecycle Management
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

https://servicenet.t-systems.com/pdm-workbench

☎ +49 (0)40 5395 2020
✉ +49 (0)3915 80125688

mail : *cmi_support@t-systems.com*

Manual History

Version	Date	Version	Date
1.0	April 2005	3.0	October 2011
2.0	November 2006	3.1	February 2012
2.1	November 2007	3.2	March 2012
2.2	September 2008	3.3	October 2012
2.5	September 2010	3.4	April 2013
		3.5	October 2013
		3.6	April 2014

This edition obsoletes all previous editions.

Your Comments are Welcome

Please feel free to tell us your opinion; we are always interested in improving our publications. Mail your comments to:

T-Systems International GmbH
GDC Product Lifecycle Management
Fasanenweg 5
70771 Leinfelden-Echterdingen
Germany

mail: *cmi_support@t-systems.com*

Preface

About this Manual

This manual provides installation and configuration information for the PDM Workbench. Before using this guide, be sure you understand:

- the Microsoft Windows operating system
- the administration of the CATIA V5 system
- the administration of the Aras Innovator system

Related Documents

The following manuals contain information about installation, administration, usage and customization of the PDM Workbench:

Manual Title	Version
PDM Workbench Installation & Administration Manual	3.6
PDM Workbench User Manual	3.6

Trademarks

CATIA is a registered trademark of Dassault Systèmes.

Aras is a registered trademark of Aras Corporation.

Names of other products mentioned in this manual are used for identification purpose only and may be trademarks of their companies.

Table of Contents

CHAPTER 1 1

OVERVIEW 1

System and Software Requirements 1
Installation steps 1

CHAPTER 2 5

ADAPTING CATIA V5 5

Loading PWBCATV5 Software from CD-ROM 5
PWBCATV5 Installation 6
Configuring the installation 6
Testing the installation 15
Switching to the PWBSchema file suited for CAD document structures 17
PWBSchema modification 17
Setting of Environment Variables 18
Administrative lock for PDM Workbench Preferences 18

CHAPTER 3 21

PDM WORKBENCH DATA MODEL 21

Installation 21

CHAPTER 4 23

PDM WORKBENCH SERVER DLL 23

COPYING THE DLL 23

MODIFYING THE SERVER CONFIGURATION FILE 24

CHAPTER 5 25

CLIENT CUSTOMIZATION 25

Display names 25
Icons 25
Data model definition 26

CHAPTER 6 27

SERVER CONFIGURATION 27

Versioning 27
Accessibility of old file versions 27
Configuration Variables 28
Configuration Items 29

CHAPTER 7 37

CLIENT SCHEMA FILE CONFIGURATION 37

Structure of the Schema File 37
Attributes of the tag "PWBSchema": 37
NLS Support for Display Names 38
Contents of a PWBSchema XML tag 38
PDM Attributes and Form Attributes 43
PDM Objects 47
PDM Relations 51
Data Sources 53
Configurations for specific functionalities 55
Basic Drawing Link Support 55
Basic Multi-Model Link Support 55

Management of CATIA templates in Innovator *55*
Autoname Support using Innovator Sequence Items *58*
CATDrawing attribute mapping *60*
Configurable CATIA components support *60*
Default sort criteria for query results *60*
Derived viewable files *61*
Support for relating a new CATIA file to an existing Part *61*
"Delete relation" context menu action in the PDM structure window *61*
The Autoname functionality can use a server method *61*
The PE_DeleteCADStructure server event method can be removed *63*
Using CATIA V4 models *63*
Support for the new CAD structure instance handling introduced in Innovator 9.4 and 10.0 ..... *64*
Standard Part Functionality *64*
Check for CAD document CATIA release at PDM update *66*
Extended attribute mapping functionality *66*
Local Workspace Information *66*

Table of Figures

PICTURE 1: DIRECTORY STRUCTURE OF THE PDM WORKBENCH INSTALLATION FILES 6
PICTURE 2: WELCOME TO THE INSTALLATION 7
PICTURE 3: LICENSE AGREEMENT 8
PICTURE 4: CHOOSE INSTALLATION SCOPE 8
PICTURE 5: CHOOSE PDM PACKAGE 9
PICTURE 6: CHOOSE PDM PACKAGE (WITH PROPOSAL) 9
PICTURE 7: CHOOSE INSTALL LOCATION 10
PICTURE 8: CHOOSE CATIA INSTALLATION 10
PICTURE 9: CATIA V5 ENVIRONMENT FILE SELECTION 11
PICTURE 10: PWB EXCHANGE DIRECTORY 11
PICTURE 11: LOCATION OF SOAP TARGET URL 12
PICTURE 12: INSTALLATION PROGRESS 12
PICTURE 13: INSTALLATION PROGRESS END 13
PICTURE 14: INSTALLATION FINISHED 13
PICTURE 15: EVENT PROPERTIES 14
PICTURE 16: PDM WORKBENCH TOOLBAR BEFORE THE LOGIN 15
PICTURE 17: THE PDM WORKBENCH TOOLBAR AFTER THE LOGIN 15
PICTURE 18: CATIA V5 GENERAL→GENERAL SETTINGS 16
PICTURE 19: CATIA V5 GENERAL→DOCUMENT SETTINGS 16
PICTURE 20: PDM SESSION CONFIGURATION DIALOG 17
PICTURE 21: PDM WORKBENCH PREFERENCES (ADMINISTRATOR VIEW) 18
PICTURE 22: PDM WORKBENCH PREFERENCES (USER VIEW) 19
PICTURE 23: PDM ARAS INNOVATOR IMPORT UTILITY 21
PICTURE 24: ITEM TYPE "PART" 27
PICTURE 25: ITEM TYPE "CAD" 28
PICTURE 26: ARAS INNOVATOR SERVER CONFIGURATION VARIABLES 28
PICTURE 27: PWB CONFIGURATION ITEM IN ARAS INNOVATOR 29
PICTURE 28: ARAS INNOVATOR SERVER CONFIGURATION VARIABLES 29
PICTURE 29: STANDARD ATTRIBUTES IN THE "PROPERTIES" DIALOG 32
PICTURE 30: CONFIGURATION OF STANDARD ATTRIBUTES IN ARAS INNOVATOR 32
PICTURE 31: STANDARD ATTRIBUTES IN THE "PROPERTIES" DIALOG OF THE PDM NODE 33
PICTURE 32: STANDARD ATTRIBUTES IN ARAS INNOVATOR WINDOW 33
PICTURE 33: CONFIGURATION OF USER-DEFINED ATTRIBUTES IN ARAS INNOVATOR 34
PICTURE 34: USER-DEFINED ATTRIBUTES IN THE "PROPERTIES" DIALOG OF THE PDM NODE 34
PICTURE 35: USER-DEFINED ATTRIBUTES IN ARAS INNOVATOR WINDOW 35
PICTURE 36: USER-DEFINED ATTRIBUTES IN THE "PROPERTIES" DIALOG 35
PICTURE 37: SINGLE LINE EDITOR WIDGET, UPDATE MODE 45
PICTURE 38: SINGLE LINE EDITOR WIDGET, OUTPUT MODE 45
PICTURE 39: MULTI LINE EDITOR WIDGET, UPDATE MODE 45
PICTURE 40: COMBO BOX WIDGET, SELECT MODE 45
PICTURE 41: SINGLE CHECK BOX WIDGET, SELECT MODE 46
PICTURE 42: CHECK BOXES WIDGET, SELECT MODE 46
PICTURE 43: RADIO BUTTONS WIDGET, SELECT MODE 46
PICTURE 44: SINGLE SELECTOR LIST WIDGET, SELECT MODE 46
PICTURE 45: PDM NODE IN PWB WINDOW 48
PICTURE 46: TOOLTIP OF PDM NODE IN PWB WINDOW 48
PICTURE 47: SELECT PDM OBJECT TYPE IN PDM QUERY DIALOG 49
PICTURE 48: CONTEXT ACTIONS FOR THE TYPE /PART/ASSEMBLY 50
PICTURE 49: RELATION ICON WITH RELATIONSHIP AND DESCRIPTION ATTRIBUTE 53
PICTURE 50: TEMPLATE FILE ADMINISTRATOR CONFIGURATION VARIABLE 56
PICTURE 51: EXAMPLE TEMPLATE FILE ADMINISTRATOR IDENTITY 56
PICTURE 52: PWB CONFIGURATION VARIABLES 56
PICTURE 53: CREATE DIALOG CONTAINING "IS TEMPLATE" CHECKBOX 56
PICTURE 54: TEMPLATE FILE CREATION ERROR MESSAGE 57
PICTURE 55: SAMPLE SEQUENCE ITEM 59
PICTURE 56: SEQUENCE ITEMS USED IN EXAMPLE 59

PICTURE 57: SAMPLE CATDRAWING ATTRIBUTE MAPPING CONFIGURATION..... 60
PICTURE 58: SAMPLE DERIVED VIEWABLE FILES CONFIGURATION..... 61
PICTURE 59: THE PE_DELETECADSTRUCTURE SERVER EVENT METHOD..... 63
PICTURE 60: CLASS STRUCTURE FOR TYPE 'CAD'..... 63
PICTURE 61: SAMPLE STANDARD PART ADMINISTRATOR CONFIGURATION..... 64
PICTURE 62: MAKING CATPARTS IN A LOCAL FOLDER ACCESSIBLE 65
PICTURE 63: DEFINING A CATPART AS A STANDARD PART 65
PICTURE 64: EXAMPLE PWB CONFIGURATION ATTRIBUTE MAPPING 66

CHAPTER 1

Overview

This chapter provides basic information about the installation of the *PDM Workbench*.

System and Software Requirements

Server Installation of Aras Innovator 9.4 or 10.0 on the following operation systems:

Windows Server 2008

On the CATIA client computers .NET 3.5 SP1 and MSXML4 SP3 have to be installed.

CATIA V5 Client Version CATIA V5 Client V5-6R2012, CATIA V5 Client V5-6R2013, and CATIA V5 Client V5-6R2014 on the following operation systems:

Windows 7 (64 Bit)

Important notice: CATIA V5-6R2014 SP2 is not supported.

Installation steps

This section describes which PDM Workbench modules (client and server) need to be installed.

On the client and the server two steps need to be performed each:

- Client installation: CATIA V5 Add-in (chapter 2)
- Client installation: License Manager. (For the installation of “licman20” please refer to the *Licman 2.0 Installation Manual*.)
- Server installation: PDM Workbench data model and server methods (chapter 3)
- Server installation: PDM Workbench server DLL (chapter 4)

In addition to these steps an additional feature license for the connector needs to be requested from Aras Corporation.

Feature licenses are requested by sending an email to licenses@aras.com which includes the requested feature (“T-Systems.CATIAV5Connector”), the account (company) name, and the contact person’s name.

The Aras license team will process this request and issue the feature license activation key once the request is validated.

A list of all available feature licenses can be accessed from within Aras Innovator, using the Tools > Admin > Licenses > View Feature Tree command.

Trial license requests are submitted to licenses@aras.com the same as regular license requests, by the requesting company.

CHAPTER 2

Adapting CATIA V5

The PWBCATV5 module provided by T-Systems International GmbH extends the CATIA V5 functionality to communicate with the Aras Innovator PDM system.

You should perform the following steps with your CATIA system administrator.

The **PWBCATV5_Rxx_xxVxx** module includes all of the supported platform data in a compressed file. Thus, you should choose an installation location for all CATIA V5 clients.

In the following example sections it is supposed that the software will be installed within the directory **C:\Program Files\T-Systems\PWBCATV5_Rxx_xxVxx_Aras_xx** on Windows but you can surely choose any other destination for the module.

Within the installation you will need to supply the PDM specific installation package. The file name follows the naming convention **PWBCATV5_xxVxx_Aras_xx**. Where the substring _xxVxx_ matches the corresponding substring of the CATIA module name.

Loading PWBCATV5 Software from CD-ROM

Windows XP/7

Use the Windows Explorer to locate the D**:\pwbcatv5\PWBCATV5_[Rxx]_[xxVxx].tar.Z** file on the CD. Extract the content of the archive file to a temporary installation location.

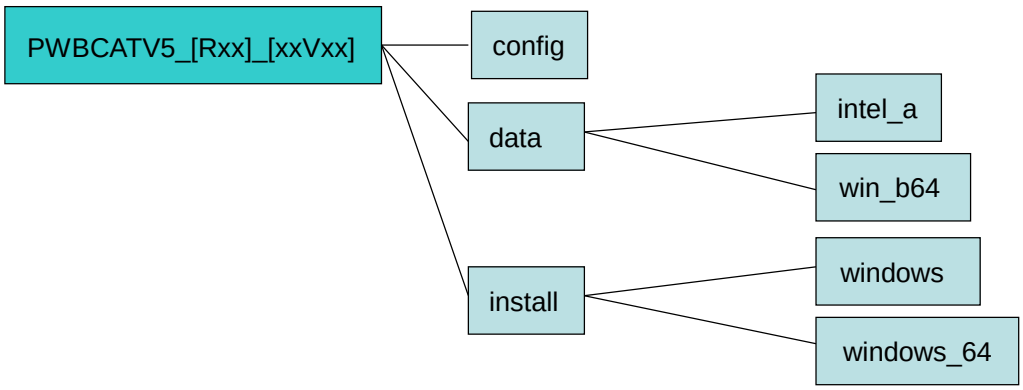
☞ **Caution: WinZip™ versions before 8.0 do not support the tar file correctly. We recommend to use WinZip™ 8.1 or above.**

PWBCATV5 Installation

After you have successfully transferred the installation files to your installation host; the following steps will install the files and configure your installation.

Configuring the installation

The **PWBCATV5_[Rxx]_[xxVxx]** Installation Directory has the following structure:



Picture 1: Directory structure of the PDM Workbench installation files

The **config** directory contains readme files and special files needed by the installer or the installed program.

The **data** directory contains the binary distributions for the PWBCATV5 module for the supported operating system mnemonics.

The supported operation systems and their mnemonics are:

Windows 7 (32 Bit)	intel_a
Windows 7 (64 Bit)	win_b64

The **install** directory contains the sub directories **windows**, and **windows_64** with all necessary data for the installer program.

Windows 7 (64 Bit)

On **Windows 7 (64 Bit)** use the Windows Explorer to run the **setup.exe** in the directory **PWBCATV5_[Rxx]_[xxVxx]\install\windows** of the installation package if you have installed the 32 Bit version of CATIA V5.

On **Windows 7 (64 Bit)** use the Windows Explorer to run the **setup.exe** in the directory **PWBCATV5_[Rxx]_[xxVxx]\install\windows_64** of the installation package if you have installed the 64 Bit version of CATIA V5.

On **Windows Vista/Windows 7** the User Account Control (UAC) will be triggered and you will have to agree that the setup program may make changes to the computer. The installer is signed with a "**T-Systems International GmbH**" certificate to ensure its integrity and source.

The prerequisites for the client package are:

- .NET Framework Version 3.5 SP1

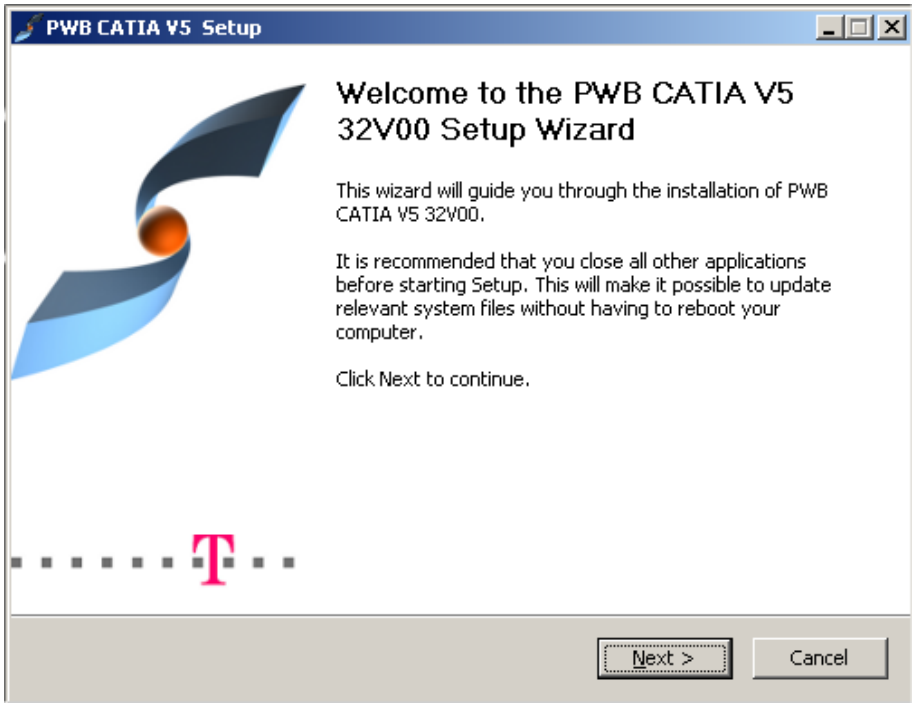
  (*https://www.microsoft.com/de-de/download/details.aspx?id=22*)

The setup will **NOT** modify the native installation of CATIA V5.

The licman20 license manager has to be installed on the CATIA V5 client host. For the installation of the license manager please refer to the *Licman 2.0 Installation Manual*.

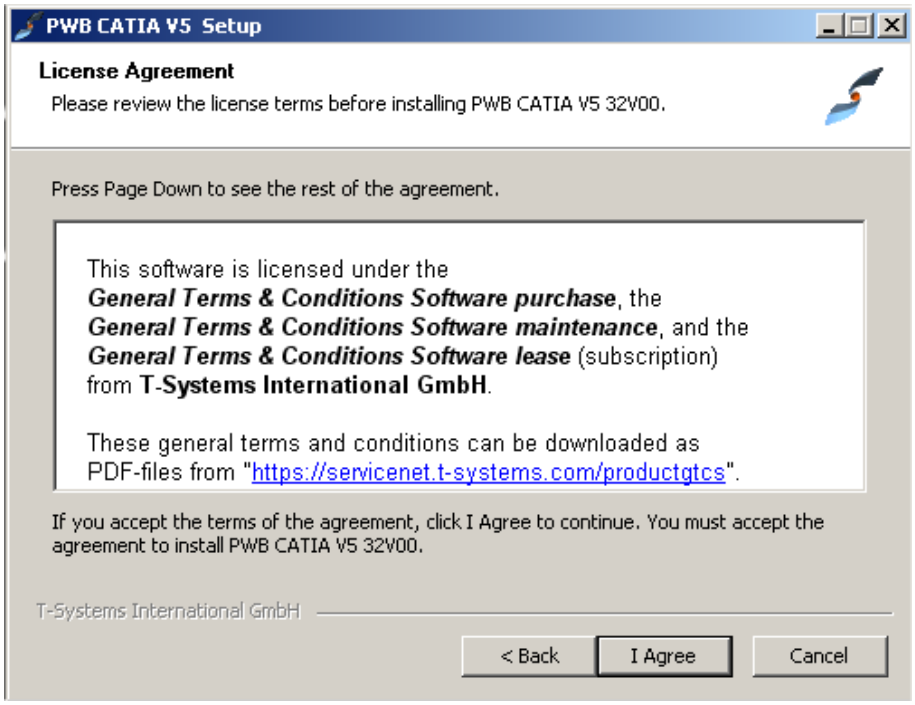
In the following the setup is shown step-by-step.

Installation process:



Picture 2: Welcome to the Installation

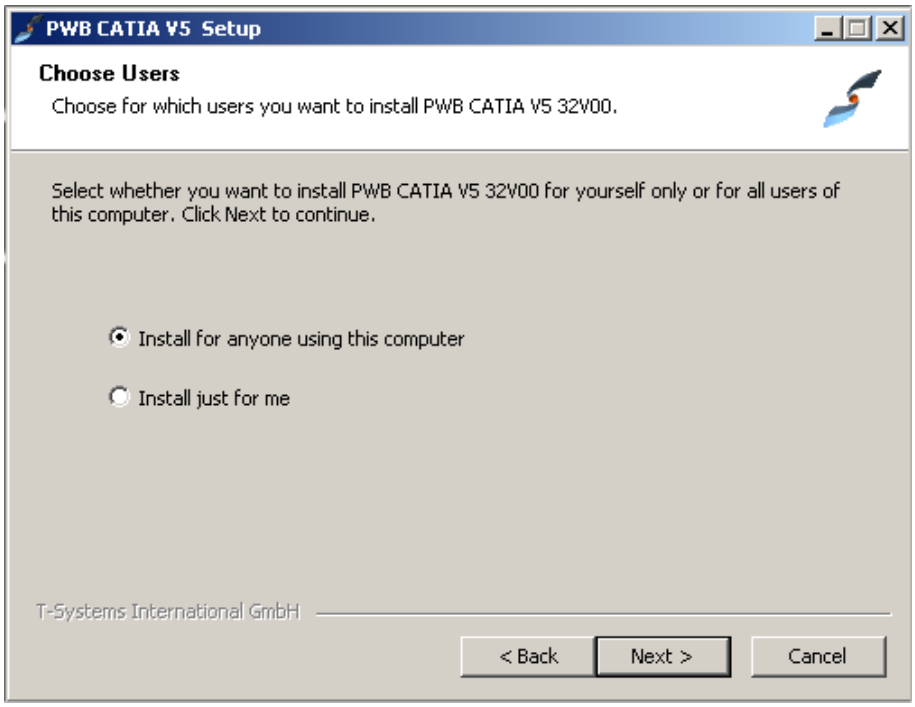
The installer software asks to approve the license terms (see *Picture 3: License Agreement*).



Picture 3: License Agreement

The installer software asks for the following input: **User scope**.

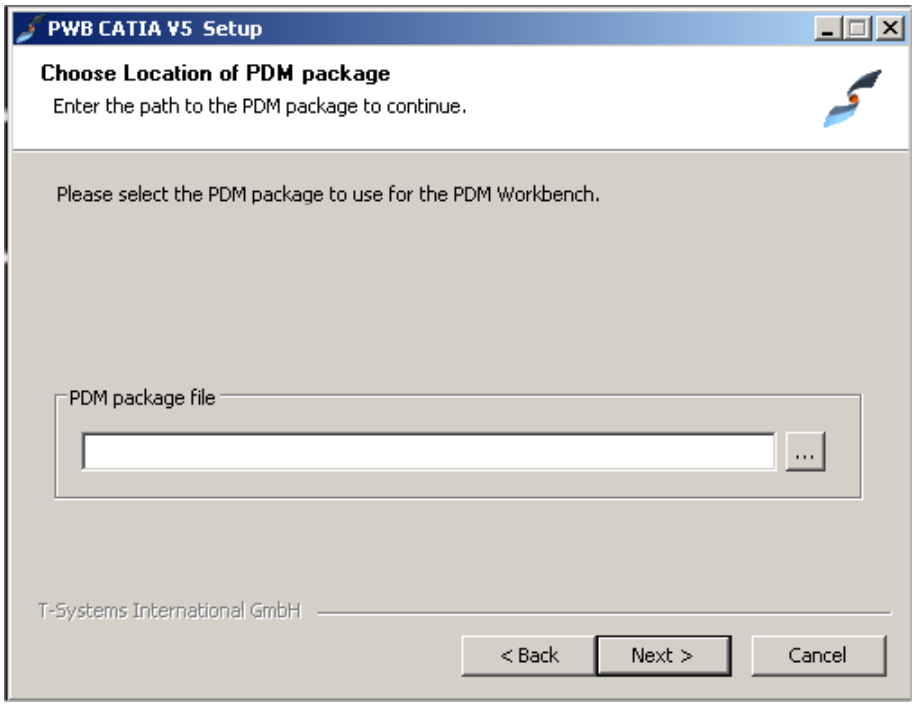
Next the installer will ask you to define the scope of the installation (see *Picture 4: Choose installation scope*). You can choose between an installation for anyone using the computer or just for the current user.



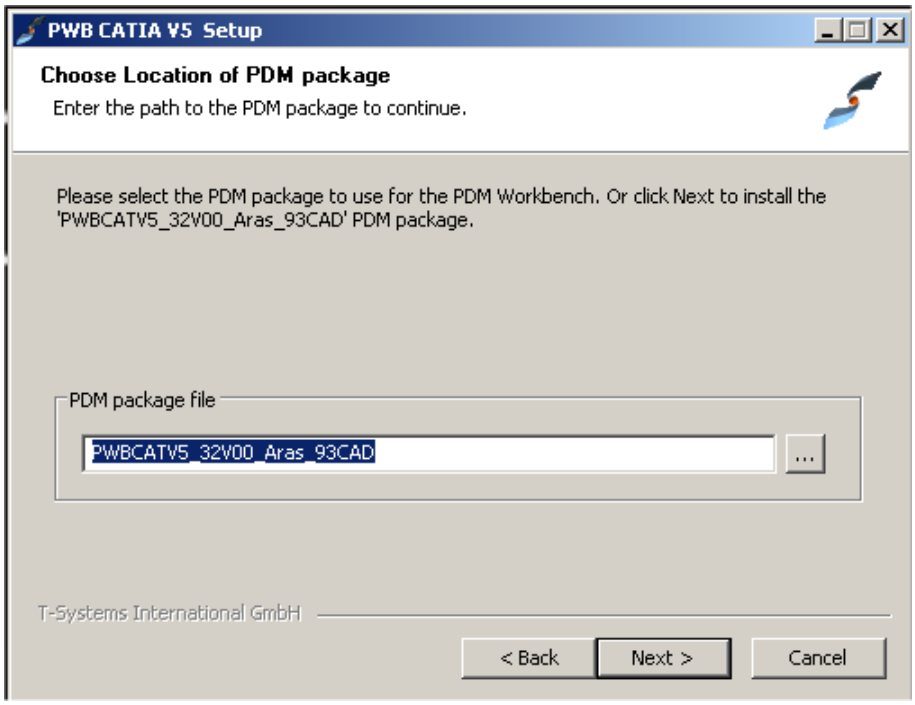
Picture 4: Choose installation scope

The installer software asks for the following input: **Location of the PDM package.**

The installer asks for the location of the PDM package to use (see *Picture 5: Choose PDM package*). If an PDM package has previously been unpacked within the installer it will be offered to install this package directly (see *Picture 6: Choose PDM package (with proposal)*).



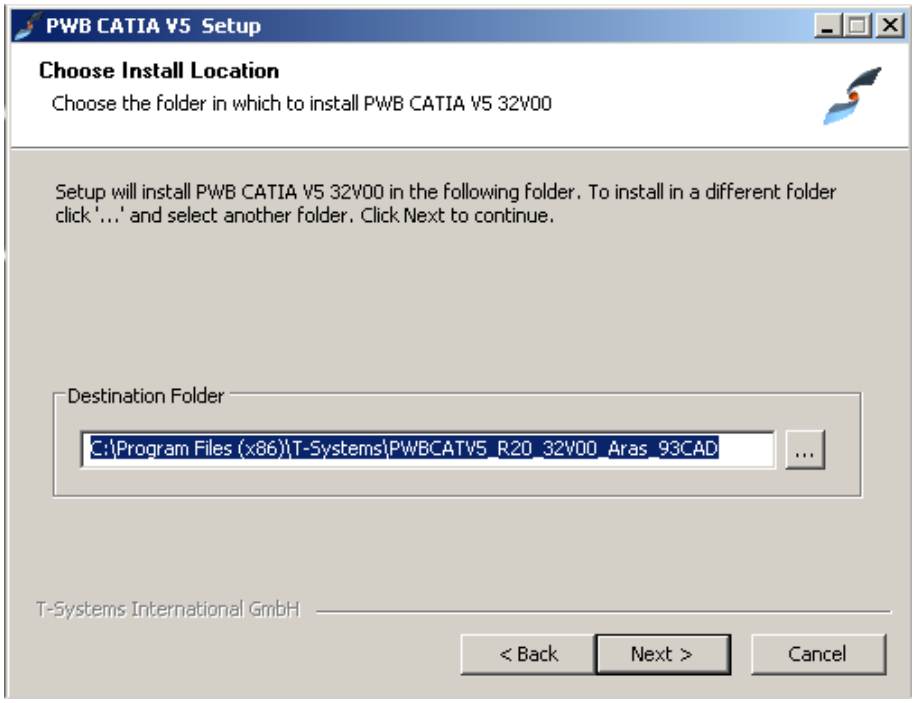
Picture 5: Choose PDM package



Picture 6: Choose PDM package (with proposal)

The installer software asks for the following input: **Installation directory.**

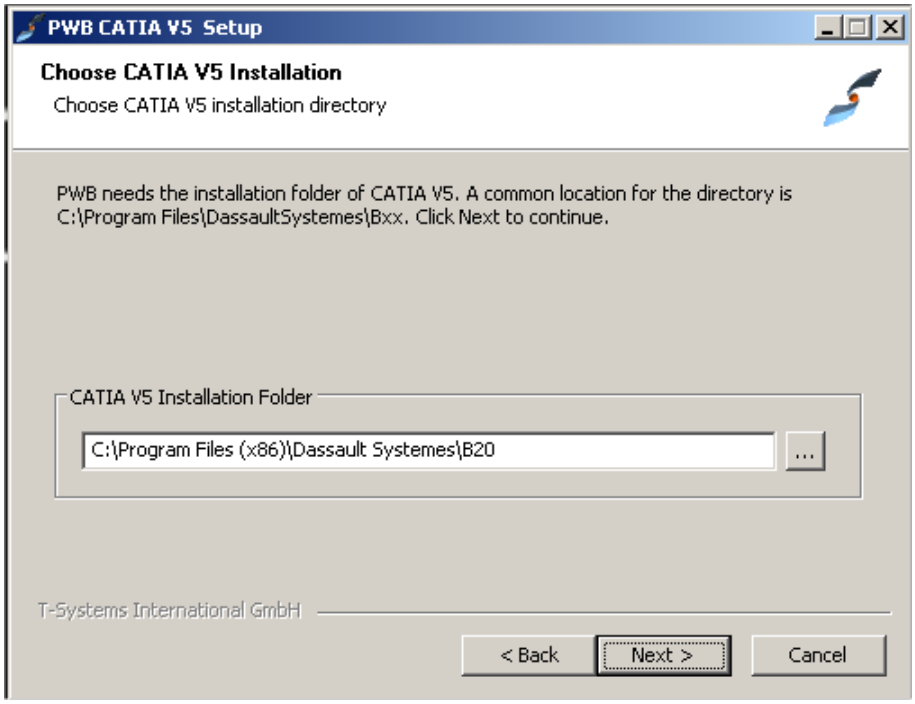
Next the installer will ask you for the target directory for the installation. You can use the given standard location or choose any other location (see *Picture 7: Choose Install Location*). The chosen folder must be empty or not existent.



Picture 7: Choose Install Location

The installer software asks for the following input: **CATIA installation directory**

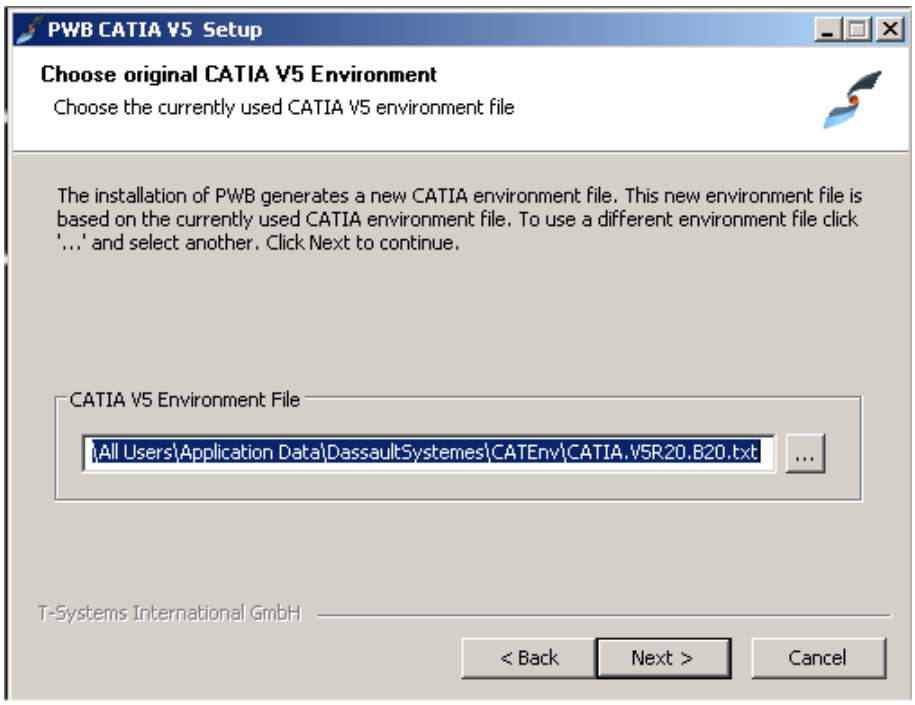
The installation path of the CATIA to use needs to be specified (see *Picture 8: Choose CATIA Installation*).



Picture 8: Choose CATIA Installation

Afterwards you will be asked for your CATIA V5 environment file (see *Picture 9: CATIA V5 Environment File Selection*).

The installation of PDM Workbench generates a new CATIA V5 environment file. This new environment file is based on the currently used CATIA V5 environment file.



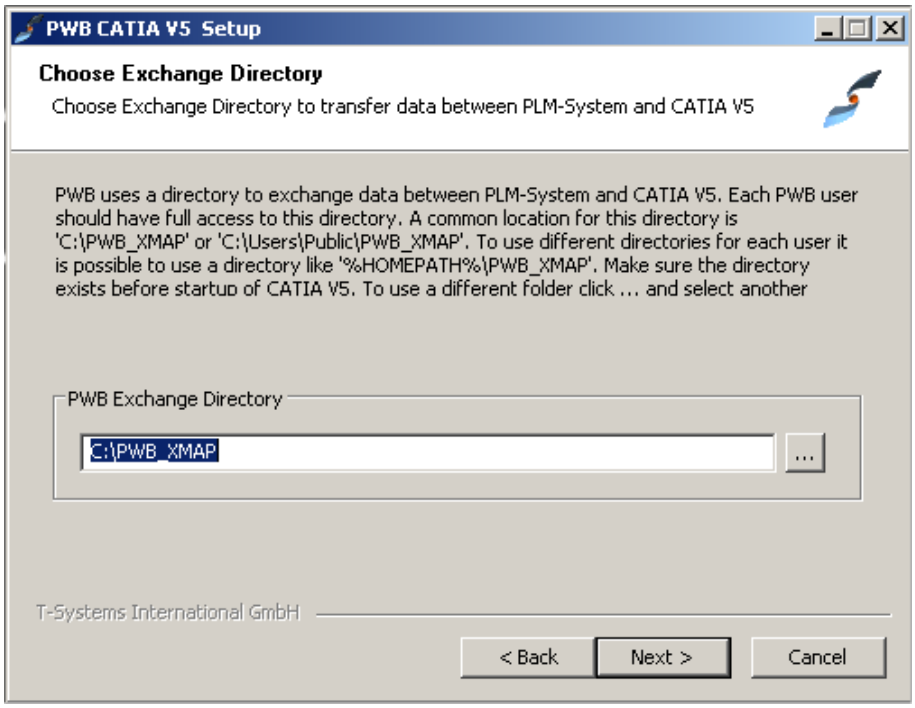
Picture 9: CATIA V5 Environment File Selection

The installer software asks for the following input: **PWB Exchange Directory**.

The PDM Workbench needs a temporary directory to perform the file transfer between CATIA and the PDM system. Make sure this directory exists for every PDM Workbench user on the CATIA client machine.

You can either use the standard location or choose any other location (see *Picture 10: PWB Exchange Directory*).

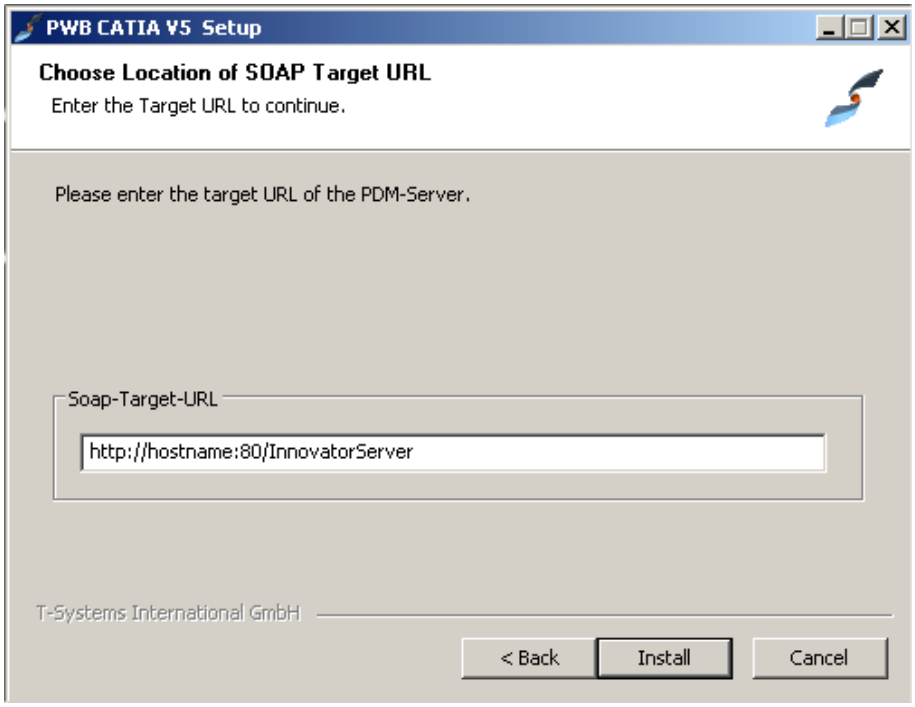
If it is planned to run more than one CATIA session at a time each session must use its own PWB Exchange Directory!



Picture 10: PWB Exchange Directory

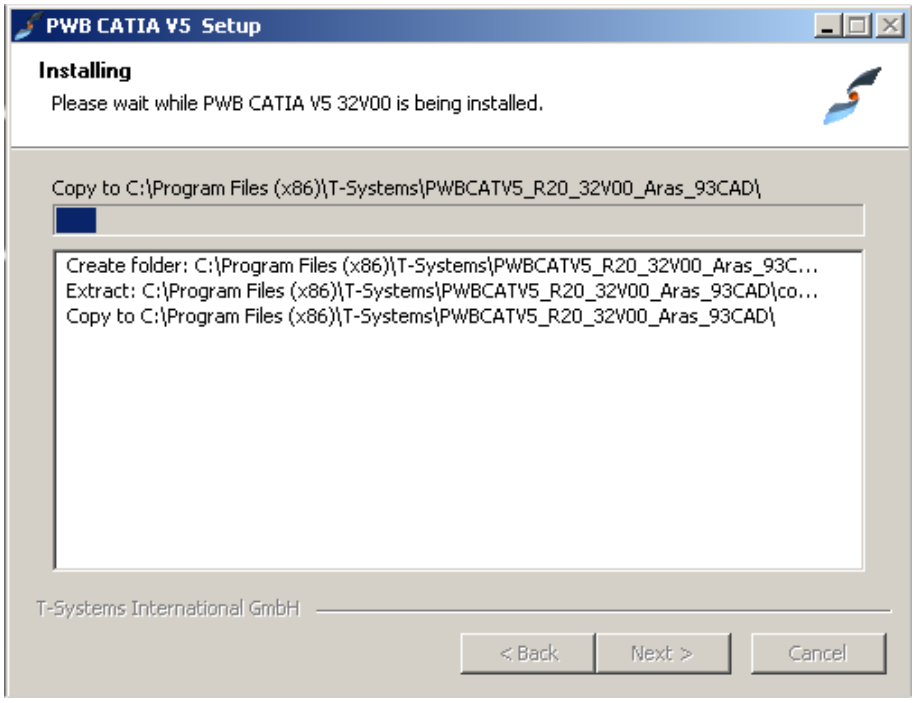
Finally you have to define the so called “Soap Target URL” for the PDM Server (see *Picture 11: Location of SOAP Target URL*).

This URL defines the host and port on which the PDM Server is reachable.

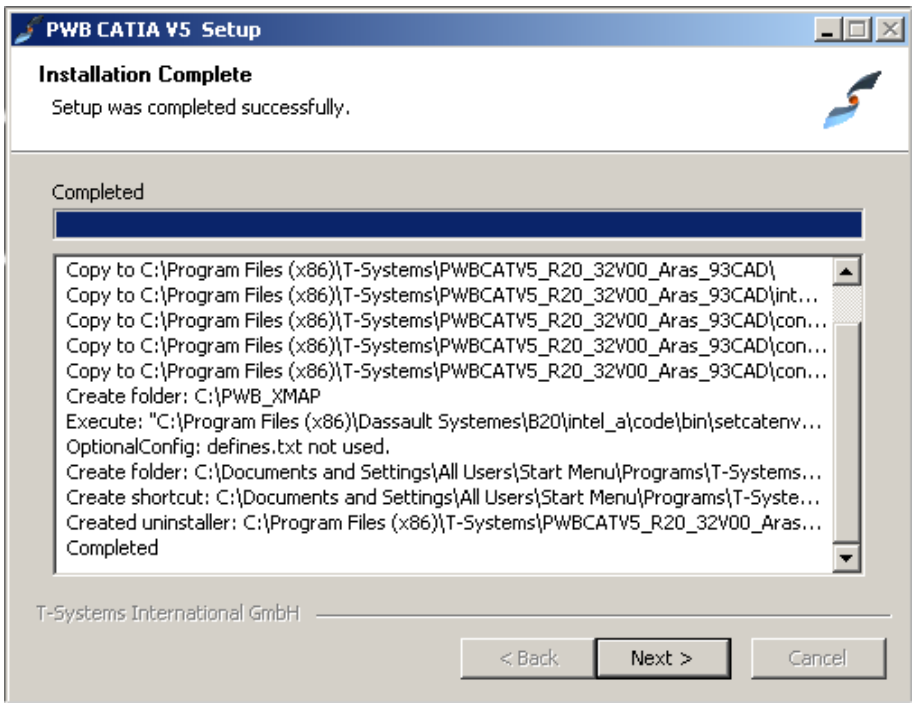


Picture 11: Location of SOAP Target URL

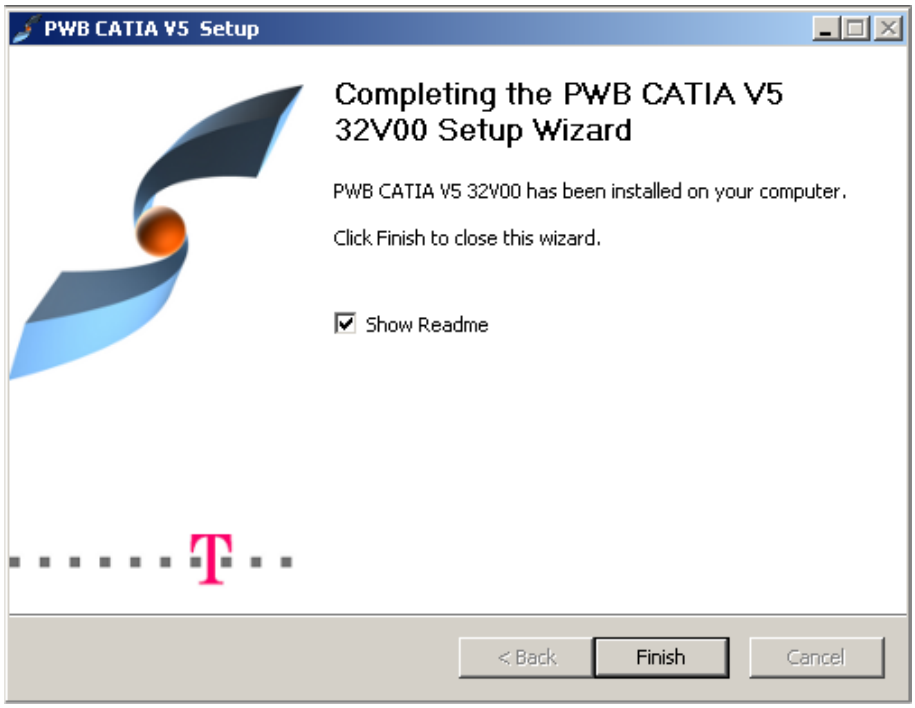
The installer will proceed in its process. The taken actions will be journalized (see *Picture 12: Installation progress* and *Picture 13: Installation progress end*).



Picture 12: Installation progress



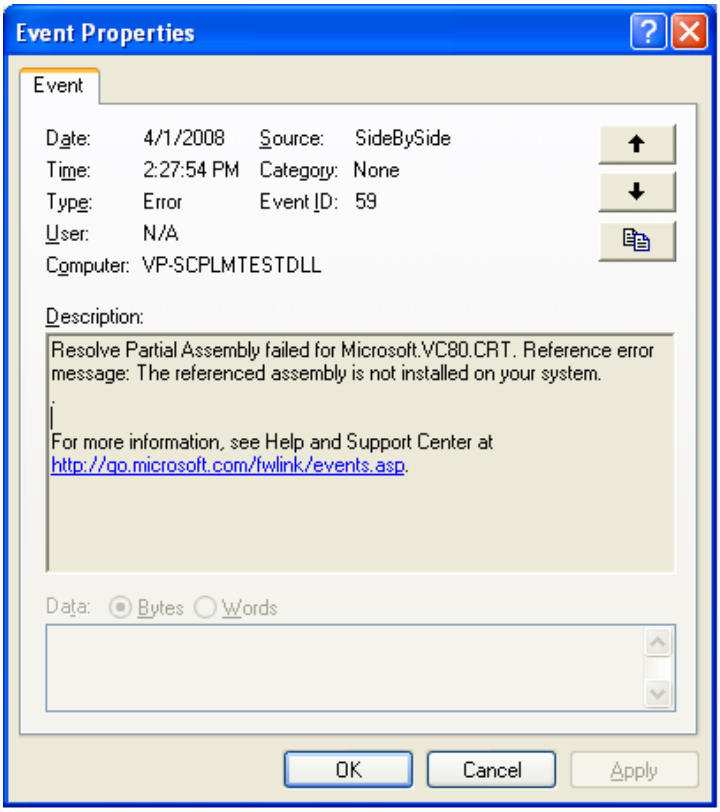
Picture 13: Installation progress end



Picture 14: Installation finished

Troubleshooting R18 (R17 (64 Bit))

If your PDM Workbench Toolbar does not appear when starting up CATIA V5, please check Windows Event Viewer → System. Check if there is a SideBySide error like this.



Picture 15: Event Properties

This error indicates that you need additional Windows runtime libraries.

You can find these runtimes in one of the following directories:

```
PWBCATV5_R17_32V00\Windows_Runtime\x64\
(required for 64 Bit CATIA only)
PWBCATV5_R18_32V00\Windows_Runtime\x64\
(64 Bit CATIA)
PWBCATV5_R18_32V00\Windows_Runtime\x86\
(32 Bit CATIA)
```

There are two possibilities to install the new runtime:

Install the runtime libraries into the Windows XP installation (recommended).

This may need system privileges.

For 32 Bit PWB/CATIA you have to extract the package `vcredist_x86.zip` and execute the setup routine `vcredist_x86\vcredist_x86.exe`

For 64 Bit PWB/CATIA you have to extract the package `vcredist_x64.zip` and execute the setup routine `vcredist_x64\vcredist_x64.exe`

or

Copy the additional libraries in a sub directory of the PWB CATIA V5 installation.

For 32 Bit PWB/CATIA you have to extract the package `Microsoft.VC80_x86.zip`. Copy the folder `Microsoft.VC80.CRT` to the binary location of the PWB CATIA module: `PWBCATV5_R18_32V00\intel_a\code\bin\Microsoft.VC80.CRT`

For 64 Bit PWB/CATIA you have to extract the package `Microsoft.VC80_x64.zip`. Copy the folder `Microsoft.VC80.CRT` to the binary location of the PWB CATIA module: `PWBCATV5_R18_32V00\win_b64\code\bin\Microsoft.VC80.CRT`

Testing the installation

Windows

Use: **Start→Programs→T-Systems→PWBCATV5_R21_32V00→PWB_START** to launch CATIA V5

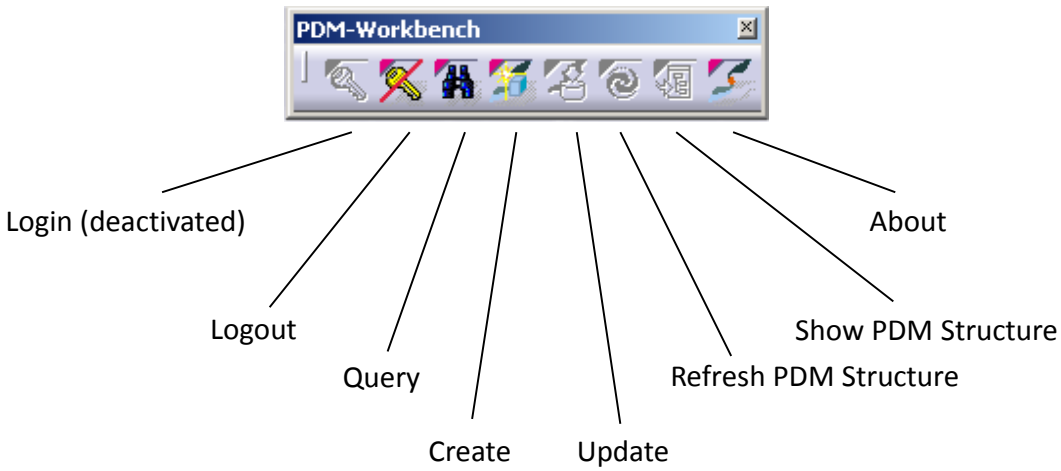
After the CATIA V5 has started the following message should appear in the command window:

```
debug on (level 1)
PDM Workbench: Module Number 1030 :
license successfully allocated
```

The License Module Number may vary.

Within CATIA V5 the following toolbar has to be visible:

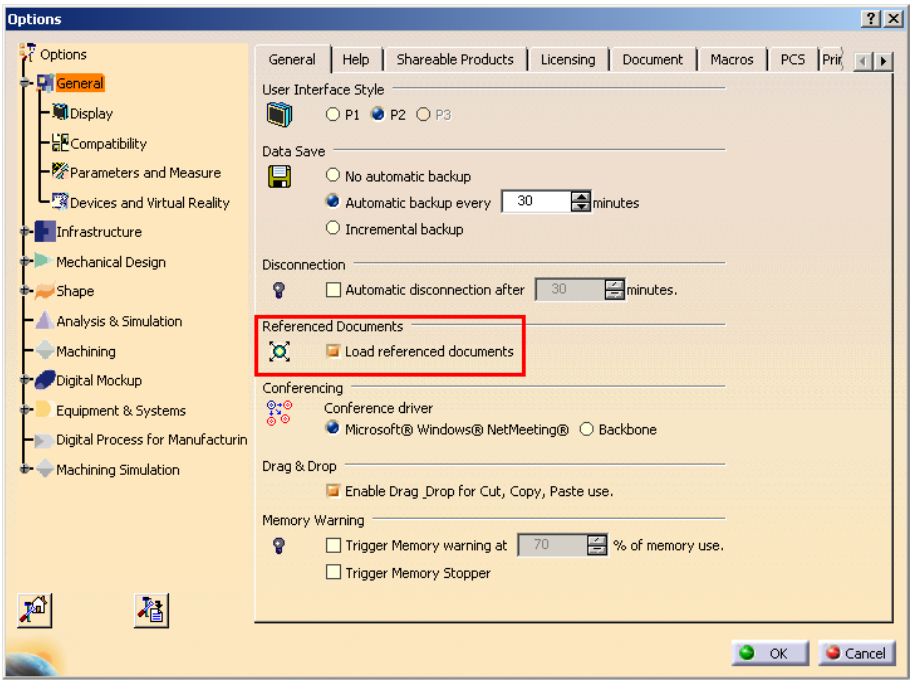
Picture 16: PDM Workbench toolbar before the login



Picture 17: The PDM Workbench toolbar after the login

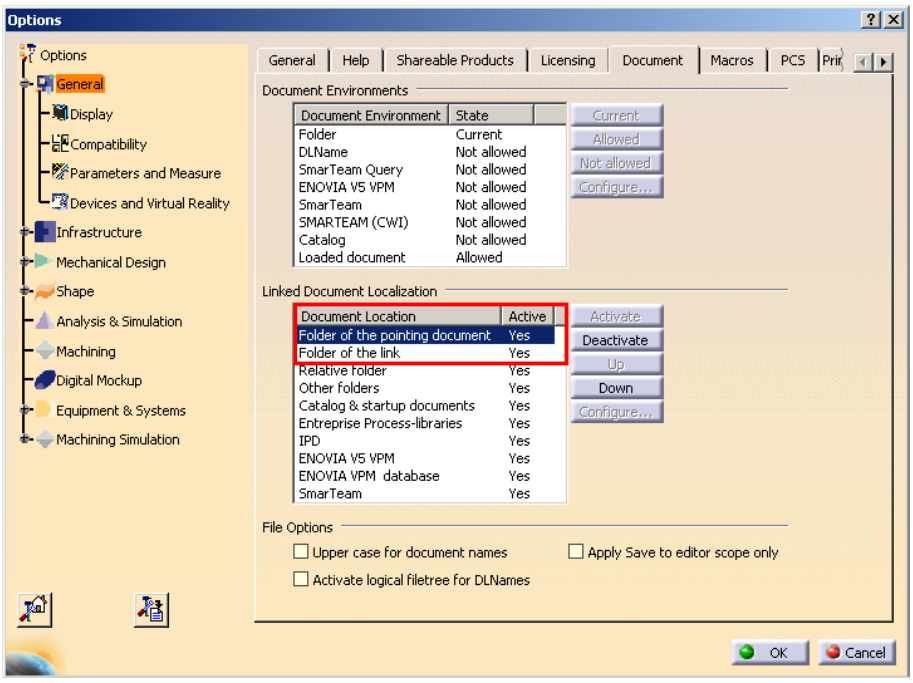
In the CATIA V5 Settings the following options have to be set as described below:

The *Load Referenced documents* option must be set in **Tools→Options→General** settings (see *Picture 18: CATIA V5 General→General Settings*).



Picture 18: CATIA V5 General→General Settings

In the *Linked Document Localization* the Options ***Folder of the pointing document*** and ***Folder of the link*** must be set to yes, and should be in this order (see *Picture 19: CATIA V5 General→Document Settings*).



Picture 19: CATIA V5 General→Document Settings

For PDM Workbench functionality please refer to the *PDM Workbench User Manual.*

Switching to the PWBSchema file suited for CAD document structures

The PDM Workbench installation package for the Aras Innovator clients also contains the PWB schema file suited to the CAD document structure mode. It is named "PWBSchema_Aras_DDM" and it resides in the 'config' directory, beside the files "PWBSchema.dtd" and "PWBSchema.xml".

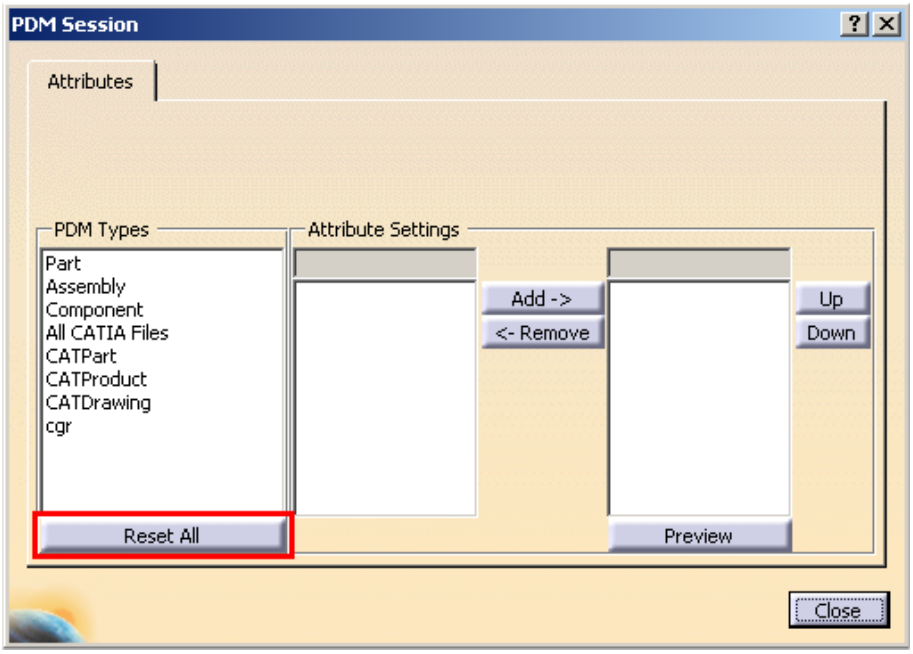
In order to use the CAD document structure mode please replace the file "PWBSchema.xml" in the installed PDM Workbench client directory (e.g. "C:\Program Files (x86)\T-Systems\PWBCATV5_R20_33V04_Aras_93SP7\config\PWBSchema.xml") with the other configuration file. Please also make sure that the PDM Workbench startup file "pwb_start.bat" contains the correct reference to the schema file (e.g. "SET PWB_SCHEMA_FILE=%PWBDIR%\config\PWBSchema.xml").

PWBSchema modification

In case of an update of the CATIA client or changes in the PDM Workbench configuration file PWBSchema.xml it is necessary to refresh the List View Column definition for all classes.

Please log in into the PDM system.

Choose in CATIA V5 *Tools →Options* and there *General – Compatibility – PDM Workbench*. Click on "Customize List View". The Configuration dialog opens (see *Picture 20: PDM Session Configuration dialog*).



Picture 20: PDM Session Configuration dialog

Please click the button "Reset All" in order to refresh the changes from the configuration file.

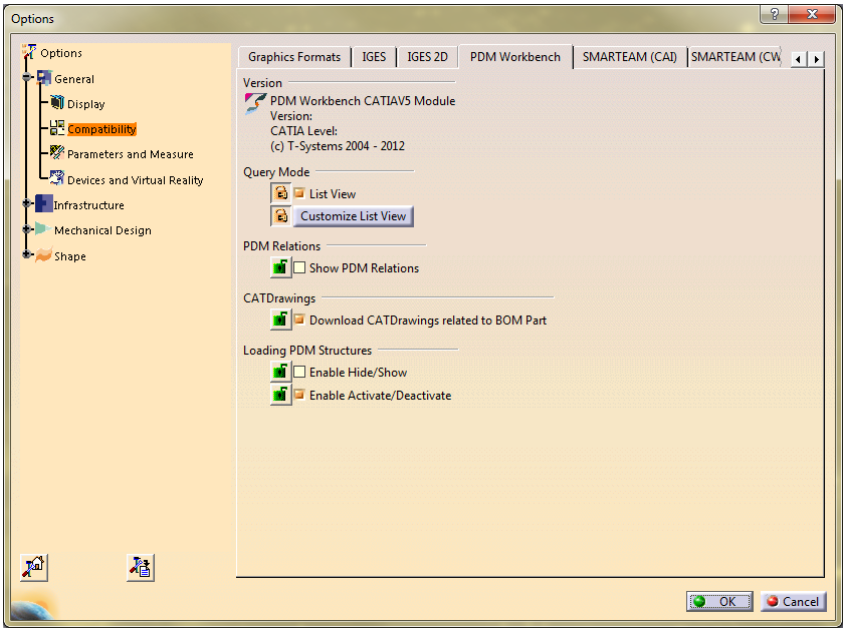
Setting of Environment Variables

The PDM Workbench software will use the following environment variables in the CATIA environment:

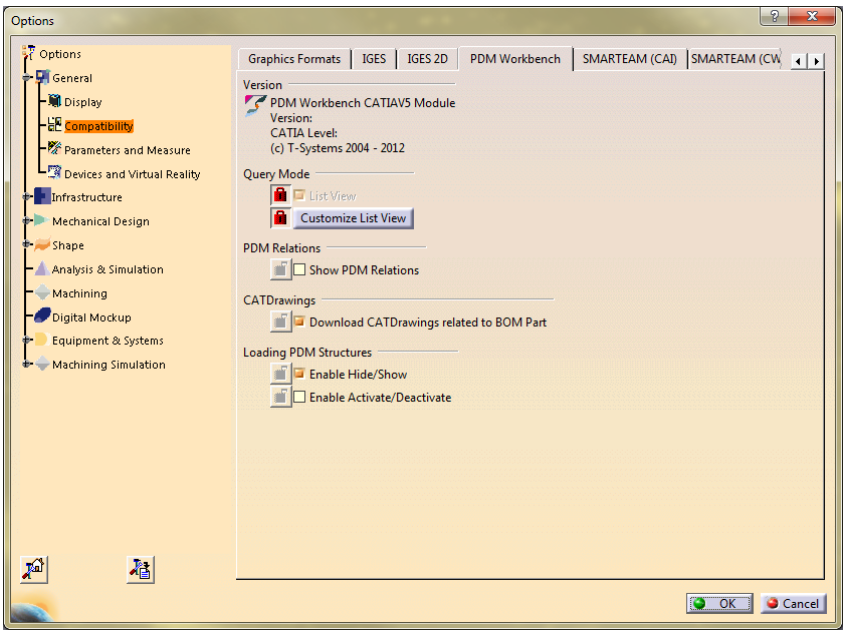
Environment variable	Comment
PWB_XMAP	The location of the exchange map directory. The exchange map directory must be unique for every started CATIA session on the same client.
PWB_SCHEMA_FILE	Path including file name of the XML configuration file.
PWB_SOAP_TARGET_URL	The URL of the web service. Host and Port, e.g. edmg119:9070
PWB_DEBUG	Set to "ON" to receive PWB debug output in the console.
PWB_ADDTEMP_PREFIX	The prefix for the rename of the Part Numbers and File Names for the "Add Temp" and "Open File Temporary" command. Default value is "TMP".

Administrative lock for PDM Workbench Preferences

An administrator can now lock the PDM Workbench preferences similar to other CATIA preferences, using the CNEXT –admin interface:



Picture 21: PDM Workbench preferences (administrator view)



Picture 22: PDM Workbench preferences (user view)

The locked preferences will have the administrator's default values and cannot be changed by the users.

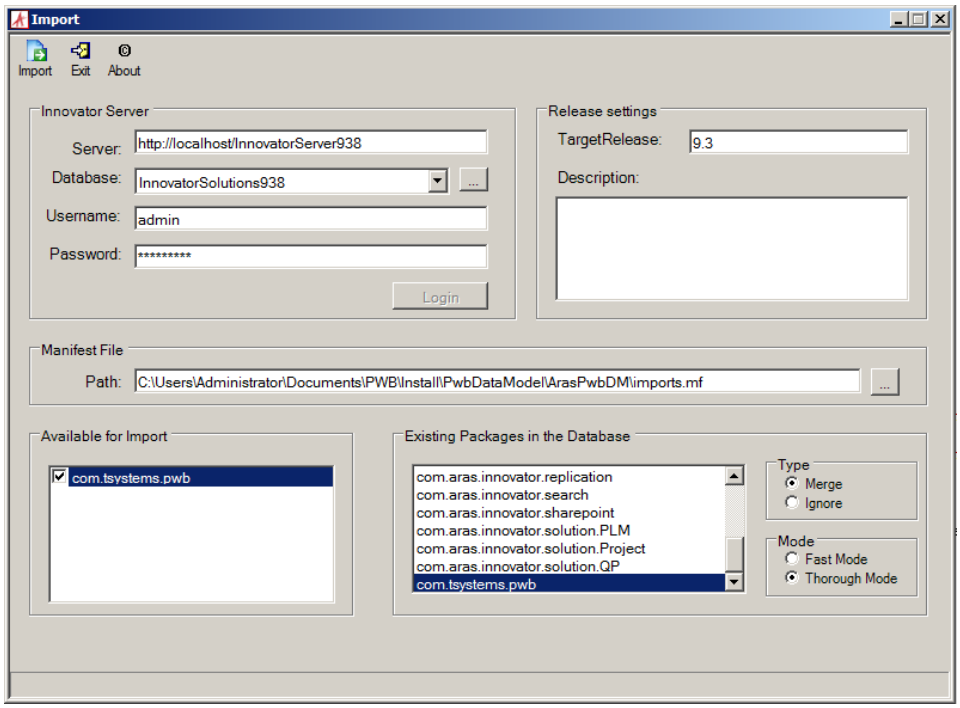
CHAPTER 3

PDM Workbench Data Model

Installation

The PDM Workbench data model and several server-side methods which call and support the main server functionality defined in the PDM Workbench server DLL (see chapter 4) need to be imported to Aras Innovator.

For this the "PwbDataModel.zip" file needs to be unpacked first. Then three packages need to be imported to Aras Innovator with the Aras Innovator import utility[1]:



Picture 23: PDM Aras Innovator import utility

Please select the manifest files

- PwbDataModel\ArasPwbDM\imports.mf
- PwbDataModel\ArasPwbDM_PLM\imports.mf

[1] The import utility has to be downloaded from the Aras homepage and to be installed. Link: www.aras.com --> Community --> Projects --> Package Import Export Utility

- PwbDataModel\ArasPwbDM_Core\imports.mf
- in the import utility in this order and perform the import (Type is "Merge", Mode is "Thorough Mode") (see *Picture 23: PDM Aras Innovator import utility*).

CHAPTER 4

PDM Workbench Server DLL

Copying the DLL

Please copy the files

```
PwbServerAddin.dll
PwbServerAddin.pdb (optional)
```

from the distribution package to the Aras Innovator server directory

```
C:\Program Files\Aras\Innovator\Innovator\Server\bin
```

or to the corresponding directory if the Aras Innovator server has been installed in a different directory.

Modifying the server configuration file

Also, please modify the file

`C:\Program Files\Aras\Innovator\Innovator\Server\method-config.xml`
by adding the highlighted lines:

```
...
<MethodConfig>
  <ReferencedAssemblies>
    <name>System.dll</name>
    <name>System.XML.dll</name>
    <name>System.Web.dll</name>
    <name>System.Data.dll</name>
    <name>$(binpath)/IOM.dll</name>
    <name>$(binpath)/InnovatorCore.dll</name>
    <name>$(binpath)/CoreCS.dll</name>
    <name>$(binpath)/SPConnector.dll</name>
    <name>$(binpath)/PwbServerAddin.dll</name>
  </ReferencedAssemblies>
...

...
<Template name="CSharp" line_number_offset="39"><![CDATA[
using System;
using System.IO;
using System.Xml;
using System.Text;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Net;
using System.Web;
using System.Web.SessionState;
using System.Globalization;
using Aras.IOM;
using PwbServerAddin;
using PwbServerAddin.Base;

namespace $(pkgname)
{
...
```

CHAPTER 5

Client Customization

The display names in the CATIA V5 workshop can be changed.

The data model of the backend PDM system has to be defined for the CATIA V5 workshop.

The variable $CATIA_INSTALL_DIR defines the installation directory of the PDM Workbench CATIA client.

Display names

The Native Language Support (NLS) files for the CATIA V5 workshop are placed in the following directory:

$CATIA_INSTALL_DIR\intel_a\resources\msgcatalog

or

$CATIA_INSTALL_DIR\win_b64\resources\msgcatalog

and the sub directories for the different languages.

There are several NLS files for the dialogs and commands.

The displays for the PDM schema file (see above) are defined in the following files:

- PWBSchemaDisplayNames.CATNls
- PWBSchemaDisplayNames_Aras_Aras.CATNls where the first “Aras” corresponds with the “system” value and the second "Aras" corresponds with the “customization” value of the PDM systems in the PDM schema file.

The displays for the error messages are defined in the following file:

- PWBUserErrors.CATNls

The displays in these NLS files can be changed.

Icons

The icons for the objects for the CATIA V5 workshop are placed in the following directory:

$CATIA_INSTALL_DIR\intel_a\resources\graphic\icons\normal

or

$CATIA_INSTALL_DIR\win_b64\resources\graphic\icons\normal

The icon names correspond to the names used in the PDM schema file.

Data model definition

The configuration of the data model for Aras Innovator has to be done in the configuration file (xml schema file) to be used by the PDM Workbench module within CATIA V5.

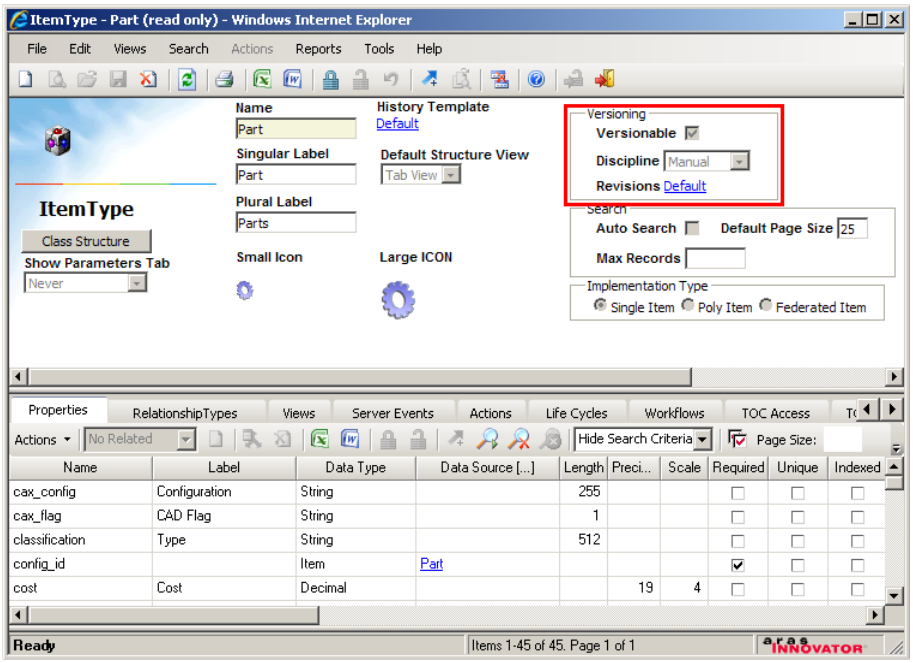
CHAPTER 6

Server Configuration

This chapter describes the configuration of the server side of the PDM Workbench integration.

Versioning

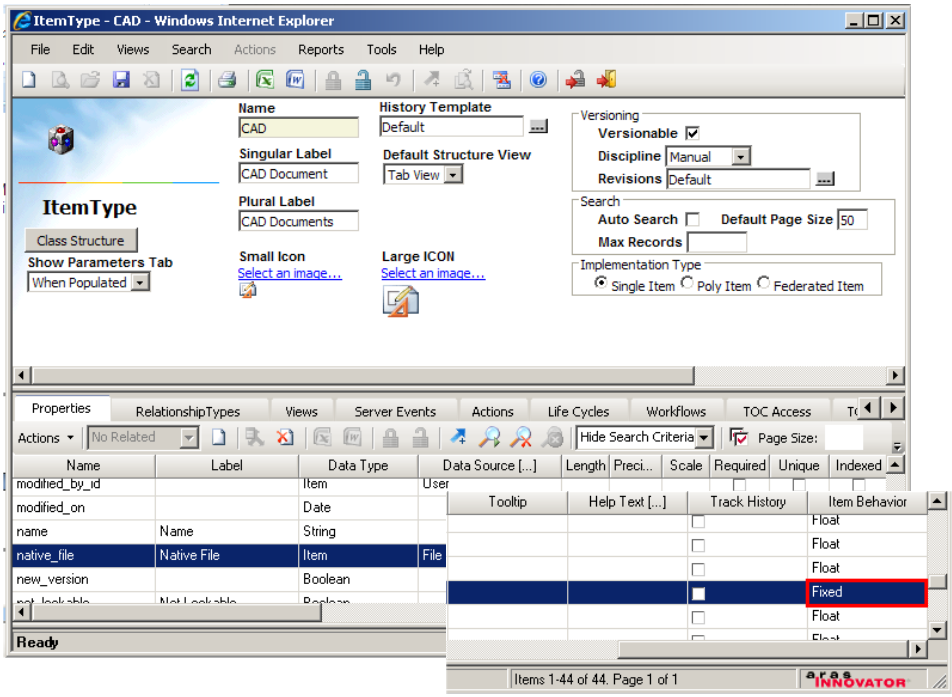
Please set the versioning discipline for the item types "Part" and "CAD" to "Manual". Now the new generation of a part and CAD document will not be created automatically in case of an update (see *Picture 24: Item Type "Part"*).



Picture 24: Item Type "Part"

Accessibility of old file versions

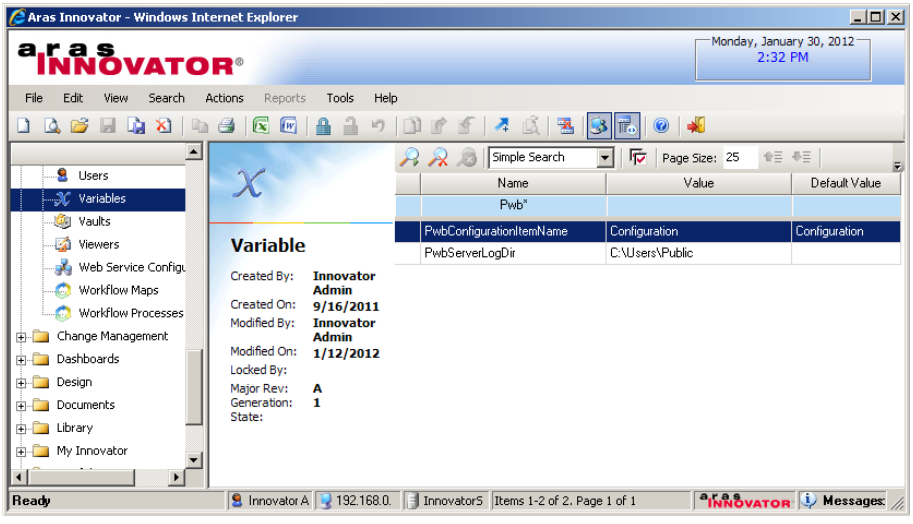
Please set the "Item Behavior" of the property "native_file" for the item type "CAD" to "Fixed". Now the old file versions can be accessed (see *Picture 25: Item Type "CAD"*).



Picture 25: Item Type "CAD"

Configuration Variables

The following Aras Innovator server configuration variables need to be set for PDM Workbench to work correctly:



Picture 26: Aras Innovator server configuration variables

- PwbConfigurationItemName

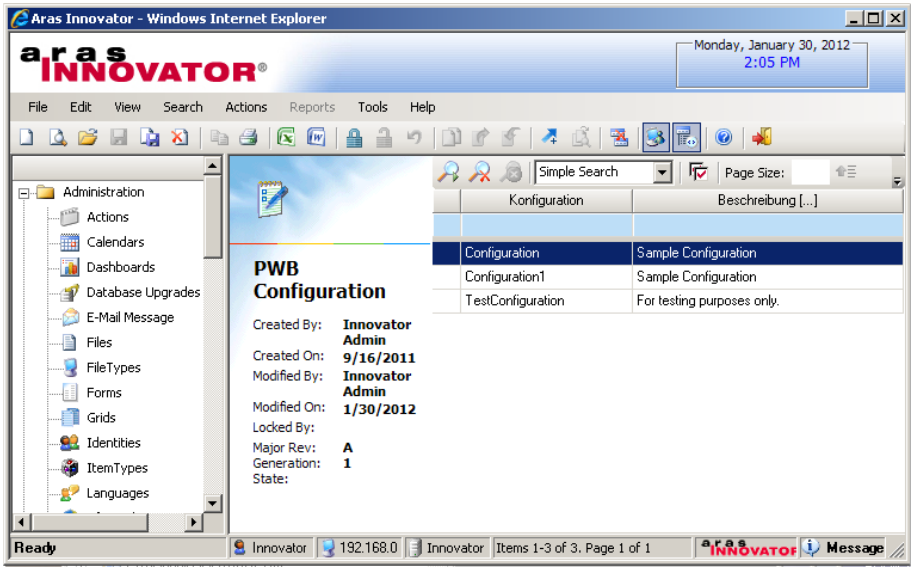
  The name of the PDM Workbench configuration item which contains additional configuration information, like the attribute mapping configuration. Please see “Configuration Items” for more details.

- PwbServerLogDir

  The absolute path of the directory into which the server log file should be written. If this variable is empty then no server log file will be written.

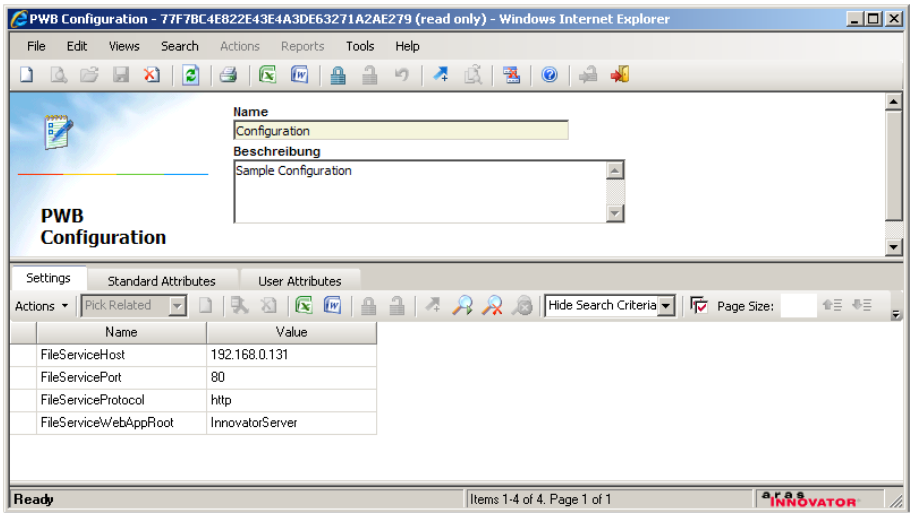
Configuration Items

In order to define the environment variables and to configure the mapping of attributes between Aras Innovator and CATIA V5 a special configuration item (see *Picture 27: PWB Configuration item in Aras Innovator*) has to be used:



Picture 27: PWB Configuration item in Aras Innovator

The following Aras Innovator server configuration variables need to be set for the PDM Workbench to work correctly:



Picture 28: Aras Innovator server configuration variables

- CadDocNumberAttr

  Default value: item_number

  Optional. The attribute containing the CAD document number of the document class.

- PartNumberAttr

  Default value: item_number

  Optional. The attribute containing the part number of the part class.

- CustomMethod_PreProcCreDlgAttrs

  Optional.

- InitialVersionCadDoc

  Optional. The initial version string for the item type "CAD Document".

- InitialVersionPart

  Optional. The initial version string for the item type "Part".

- PromoteSourceStates

  Default value: Preliminary|In Review

  Optional. A list of the promote source states, separated by "|"..

- PromoteTargetStates

  Default value: In Review|Released

  Optional. A list of the promote target states, separated by "|".

- ShowCreateDialogsDuringUpdate

  Optional. Has to be set to "true" in order to show the create dialogs during the update process.

- UseBomPartStructure

  Optional. Has to be set to "true" in order to use the "BOM Part Structure" data model. Otherwise the document data model will be used.

  .

- CreateThumbnailsFromTypes

  Default value: .CATPart|.CATDrawing

  Optional. A list of file extensions for which the thumbnails should be created by update, separated by "|".

- StandardPartAdmin

  Mandatory. Can be removed if the following server events are removed:

PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts
PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts
PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts
PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts
PwbCheckStandardPartPermissions	CSharp	6	World	Check user permissions to change standard parts

  Defines the standard part administrator identity.

-

- TemplateFileAdmin

  Mandatory. Can be removed if the following server events are removed:

PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files
PwbCheckTemplateFilePermissions	CSharp	6	World	Check user permissions to change template files

  Defines the template file administrator identity.

- QueryOrderByAttribute

  Optional. Defines an attribute by which the query results are internally ordered. This is not noticeable by the user, but it can result in significant performance improvements when a query is performaed if the attribute is in the database index.

  Example:

QueryOrderByAttribute	id

- MaxQueryResults

  Optional. Defines the maximum number of items that are retrieved in a single query. If more items exist the user is informed about that fact.

- BomPartStructureLockUnlockRelatedItem

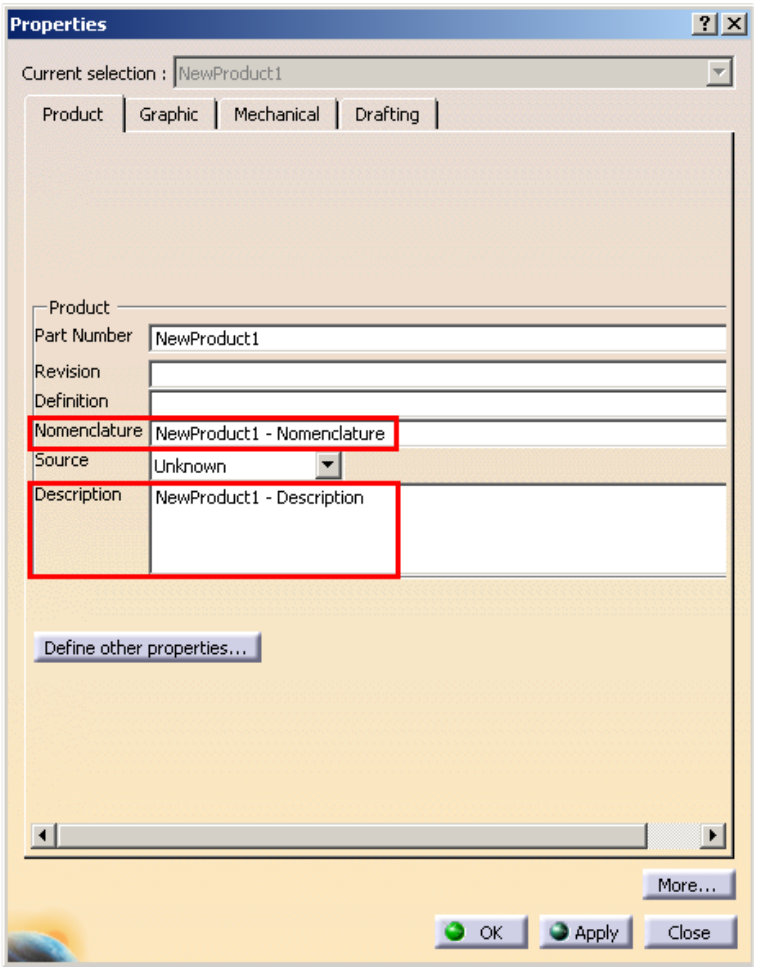
  Optional. Possible values: “part”, “cad”, or “both”.

  Normally the “Lock” and “Unlock” context menus in the CATIA window lock and unlock only the CAD document. Also locking and unlocking a part or a CAD document in the PDM structure window only locks the item where the action was performed on. With this configuration setting this behavior changes such that locking a part also locks its related CAD documents, or locking a CATPart or CATProduct CAD document locks its related part, or both of these actions are activated. So, for instance, if this setting is set to “part” or to “both” then in the BOM structure mode locking a CAD document, either explicitly in the PDM structure window, or in the CATIA window, also locks the corresponding part item.

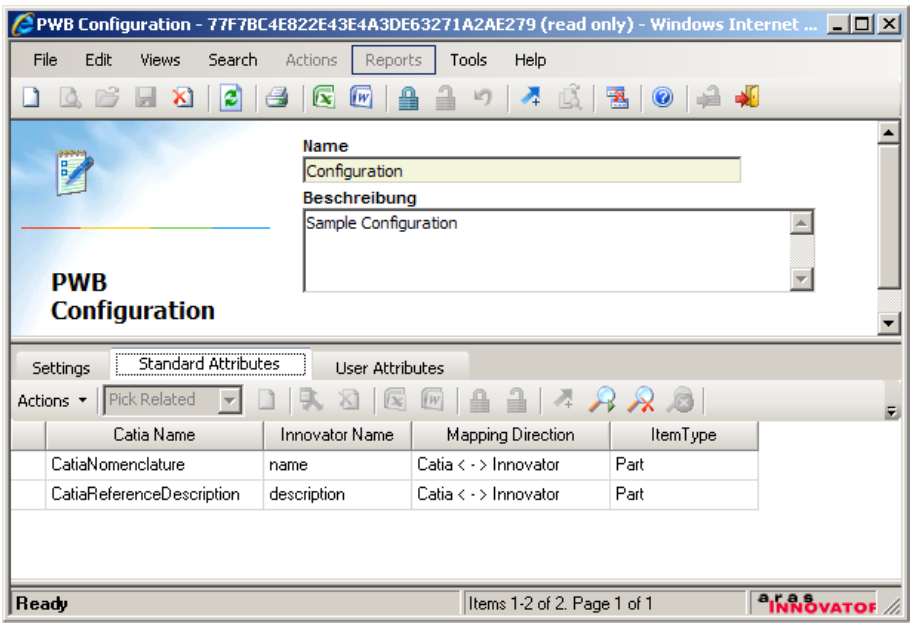
Here is a description of how to configure the attribute mapping:

CATIA standard and user-defined properties can be mapped to PDM attributes.

In the following example the standard CATIA attributes “Nomenclature” and “Description” are mapped to the attributes “name” and “description” of the Aras Innovator part object (see *Picture 29: Standard attributes in the “Properties” dialog* and *Picture 30: Configuration of standard attributes in Aras Innovator*).

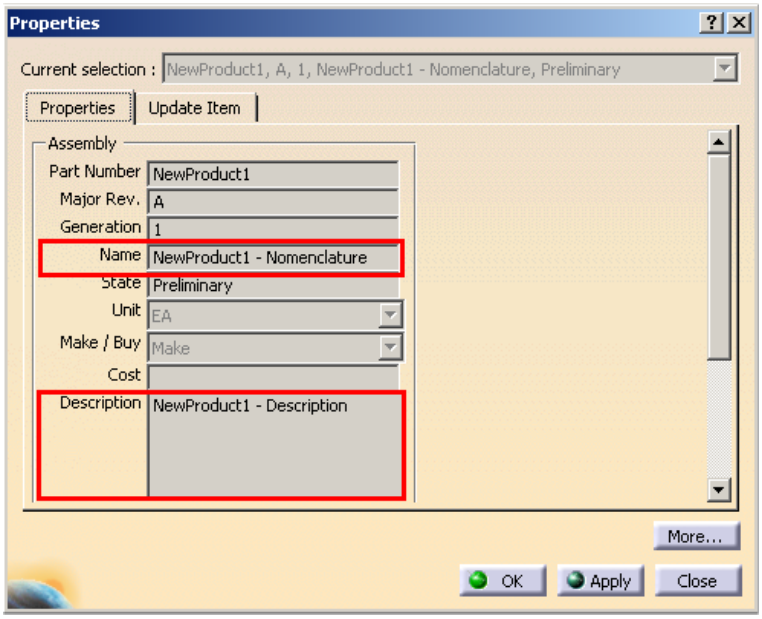


Picture 29: Standard attributes in the "Properties" dialog

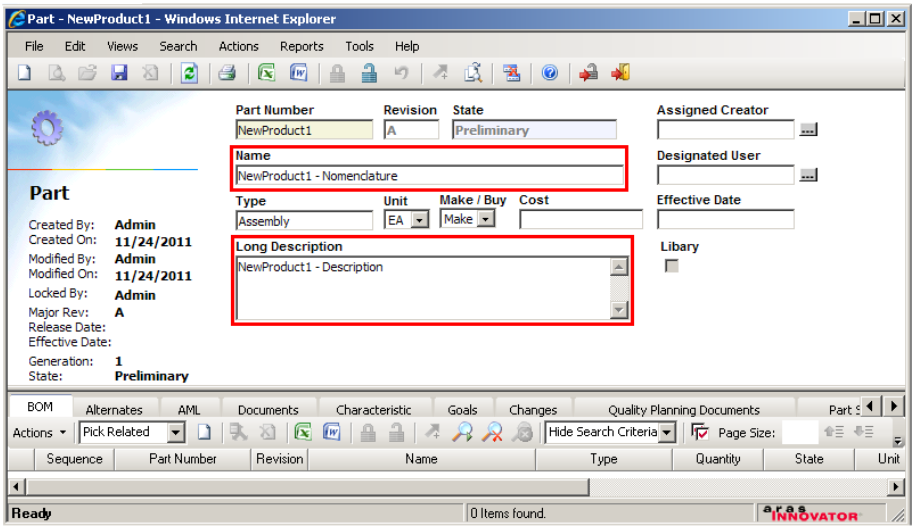


Picture 30: Configuration of standard attributes in Aras Innovator

After creating the part with Update the defined CATIA attribute values have been written to the PDM part object (see *Picture 31: Standard attributes in the "Properties" dialog of the PDM node* and *Picture 32: Standard attributes in Aras Innovator window*).

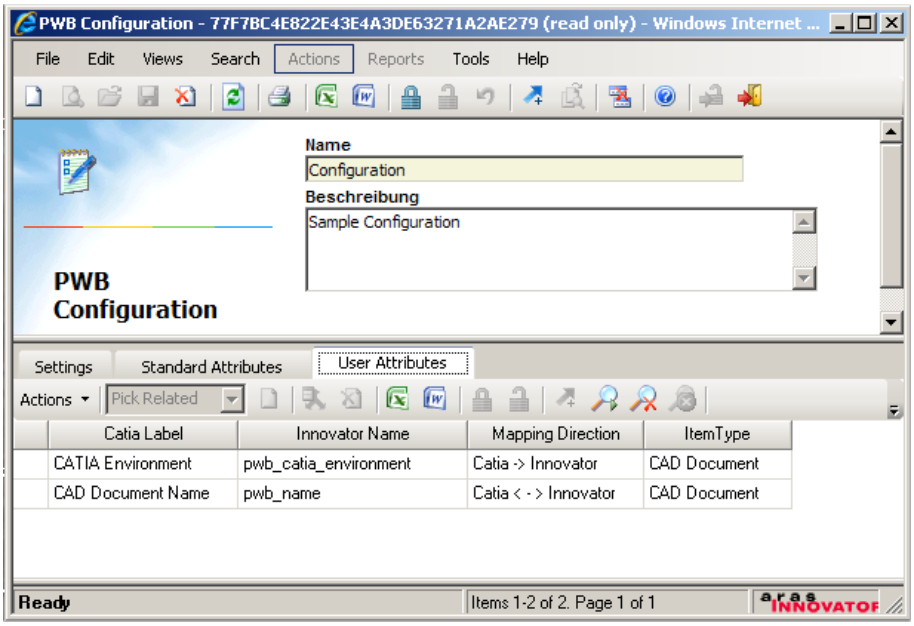


Picture 31: Standard attributes in the "Properties" dialog of the PDM node



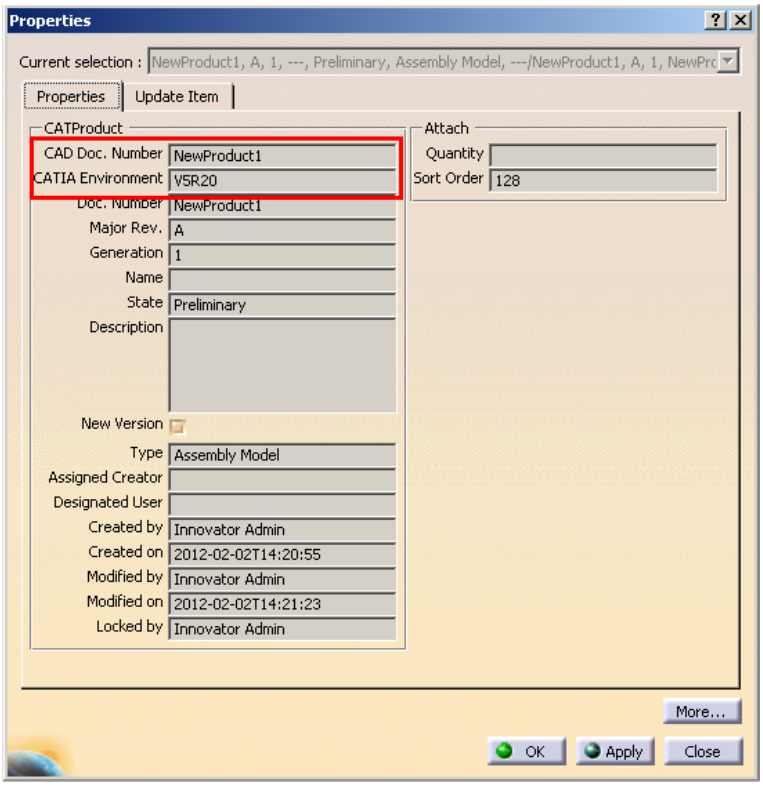
Picture 32: Standard attributes in Aras Innovator window

User-defined CATIA properties can also be mapped (see *Picture 33: Configuration of user-defined attributes in Aras Innovator*).

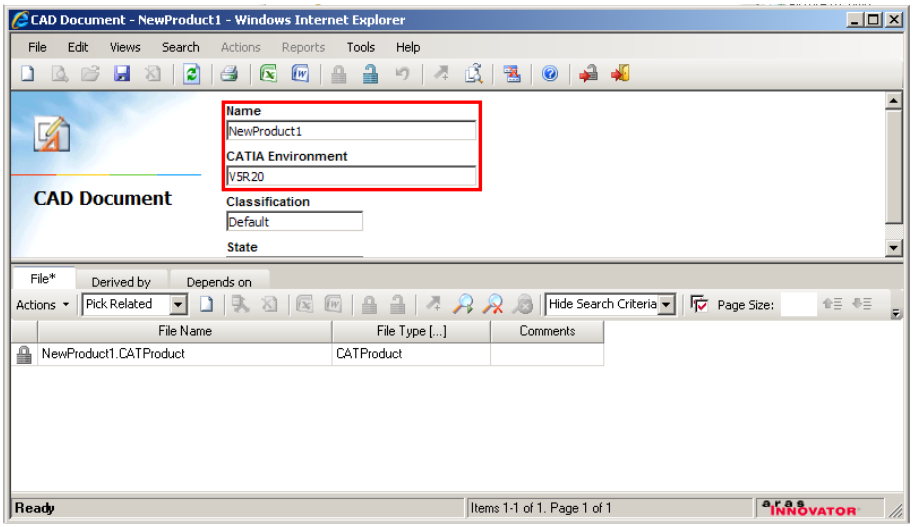


Picture 33: Configuration of user-defined attributes in Aras Innovator

While the structure is imported the values are written to the defined attributes of the Aras Innovator CAD document object (see and *Picture 34: User-defined attributes in the "Properties" dialog of the PDM node* and *Picture 35: User-defined attributes in Aras Innovator window*).

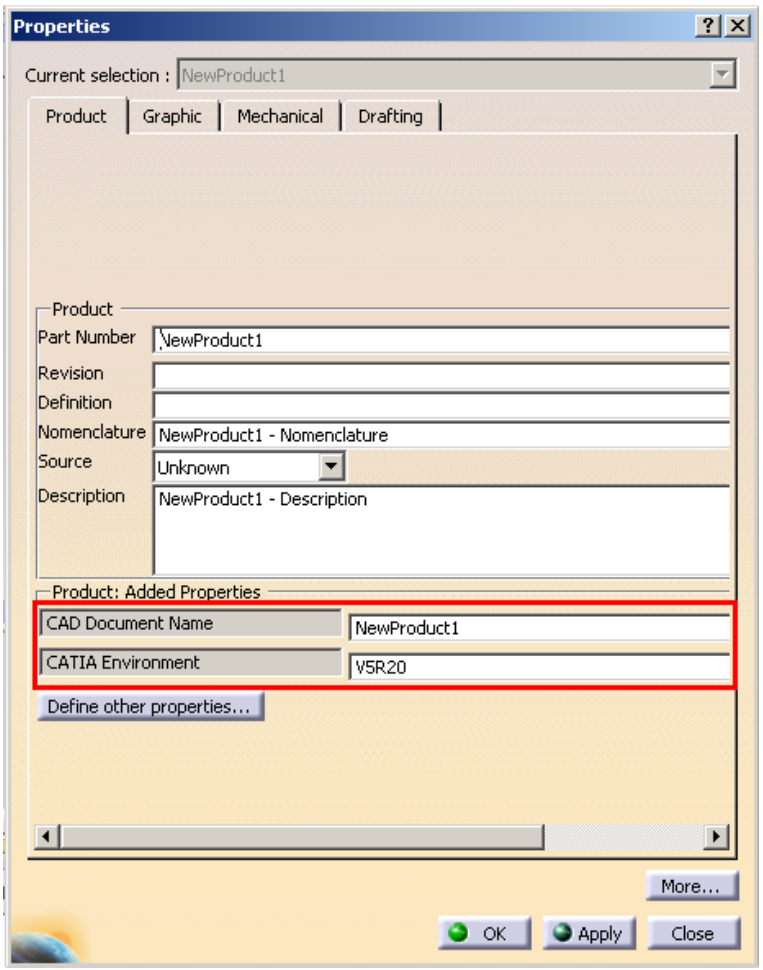


Picture 34: User-defined attributes in the "Properties" dialog of the PDM node



Picture 35: User-defined attributes in Aras Innovator window

After the import or after loading the structure it can be shown that the values are written from the PDM attributes into the CATIA files (see *Picture 36: User-defined attributes in the "Properties" dialog*).



Picture 36: User-defined attributes in the "Properties" dialog

CHAPTER 7

Client Schema File Configuration

This chapter describes the configuration of the client side of the PDM Workbench integration.

Structure of the Schema File

The main purpose of the PDM Workbench Schema File is to define which subset of the objects, relations, and attributes in the PDM system should be made available to the design engineer who is working with CATIA V5 and who needs to save the CATIA files he is working on in a PDM system.

The classes of PDM objects that the user can query, create, etc. will be defined in the Schema file, as well as the dialogs which contain these objects' attributes and the PDM relations which relate the PDM objects to each other.

The Schema file can be edited with a text editor, or a XML editor.

At the root of the Schema XML file, there is the tag "PWBSchemata". Its child tags are named "PWBSchema". The information about every PDM system that can be accessed is defined inside this "PWBSchema" tag. There is one "PWBSchema" tag for every PDM system and every PDM system customization that can be accessed from the PDM Workbench.

```
<!-- root tag -->
<PWBSchemata>
   <!-- out-of-the-box Aras -->
   <PWBSchema system="Aras" customization="Aras"
              displayName="NLS_System" visibleLength="15">
   ...
   </PWBSchema>

   <!-- customization of Aras -->
   <PWBSchema system="Aras" customization="PDM-Customization"
              displayName="NLS_System" visibleLength="15">
   ...
   </PWBSchema>
</PWBSchemata>
```

Attributes of the tag "PWBSchema":

- "system" — Contains the short name of the PDM system. Supported is "Aras" for Aras Innovator.
- "customization" — Contains the name of the customization. If the PDM system is used out of the box without any customization, then the convention is to use the short name as defined for the

"system" attribute.

- "displayName" — Contains the NLS (native language support) name of the PDM system or customization that is defined in the "PWBSchema" XML tag.
- "visibleLength" — Contains the visible length of the display name to be shown in the dialogs of CATIA V5.
- "allowedLength" — Contains the allowed length of the values inserted in the text editor widgets in characters.

NLS Support for Display Names

Many XML tags (PWBSchema, frame, language, object, relation, attribute, etc.) have an attribute with the name "displayName". The string that represents the value of that attribute defines the language-specific display name for that object that the PDM Workbench users can see. The language-specific name is defined in the files "PWBSchemaDisplayNames.CATNls" and "PWBSchemaDisplayNames_SYSTEM_CUSTOMIZATION.CATNls", where SYSTEM is the value of the "system" attribute and CUSTOMIZATION is the value of the "customization" attribute. For system="Aras" and customization="Aras" the name of the CATNls file would be "PWBSchemaDisplayNames_Aras_Aras.CATNls". That file contains the NLS names specific for that PDM system or customization, while "PWBSchemaDisplayNames.CATNls" contains the general definitions that apply to all PDM systems.

In this case, the value for the frame's display name "NLS_UserData" is defined in the file "PWBSchemaDisplayNames_Aras_Aras.CATNls":

File "**PWBSchema.xml**":

```
...
<PWBSchema system="Aras" customization="Aras" displayName="NLS_System"
           visibleLength="15">
...
```

File "**PWBSchemaDisplayNames_Aras_Aras.CATNls**":

```
...
NLS_System = "Aras Innovator";
...
```

Contents of a PWBSchema XML tag

"xmap": optional

The value contains the absolute path of the exchange map directory, where the downloaded CATIA files are stored.

Example:

```
<xmap value="C:\PWB_XMAP" />
```

If the exchange map value is defined by the CATIA V5 environment variable "PWB_XMAP", then that takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

"soapTargetUrl": optional

The value contains the URL of the server process that the PDM Workbench client uses for its SOAP requests.

Example:

```
<soapTargetUrl value="http://epdmd11:80/InnovatorServer" />
```

If the soap target URL value is defined by the environment variable "PWB_SOAP_TARGET_URL", then this one takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

"externalSoapClientCallPathEnvVar": mandatory

The name of the environment variable which contains the path to the Aras Innovator SOAP client executable.

"externalFileClientCallPathEnvVar": mandatory

The name of the environment variable which contains the path to the Aras Innovator file client executable.

"addTempPrefix": optional

The prefix for the rename of the Part Numbers and File Names for the "Add Temp" and "Open File Temporary" command.

Example:

```
<addTempPrefix value="TMP" />
```

If the add temp prefix value is defined by the environment variable "PWB_ADDTEMP_PREFIX", then this one takes precedence. The definition in the Schema file takes effect only if such a CATIA V5 environment variable does not exist.

The default value is "TMP".

"pwbWindowColor": optional

This tag contains the red, green, and blue values (0 - 255) of the PWB window background.

Example:

```
<pwbWindowColor red="143" green="155" blue="177" />
```

If not set then the CATIA V5 standard color will be used.

"maxExpansionLevel": optional

The value contains the maximum expansion level, starting from a root node the PDM Workbench will display.

Example:

```
<maxExpansionLevel value="30" />
```

If not set then all levels of the assembly will be expanded.

"colorSupersededNodesAsOutdated": optional

If this setting is set to “true” then superseded nodes (nodes where a newer generation exists) are colored magenta, like outdated nodes, where the CAD file has been modified by an other user, instead of black.

"checkForCadOwner": optional

This setting is optional, and the default value is “false”. If this setting is set to “true” then it will not be possible to load structures which have been created by a different CAD integration (‘external_owner’ attribute value is not ‘T-Systems.Mechanical.CATIA’) .

"showCreateVersionAtUpdate": optional

This setting is optional, and the default value is “true”. If this setting is set to “false” then the “Create new file versions at update?” button does not appear in the update dialog. The default value (overwriting the existing document) will always be active.

"requiredDialogAttributes": optional

XML attribute: invalidValue, example value: "*"

If this setting is active it defines values as not valid for required query dialog attributes. For instance, if the item_number attribute is defined as required, then the setting above defines the query value of "*" as invalid for that attribute.

"removeToolbarIcons": optional

This tag includes the names of the toolbar entries to be removed.

Example:

```
<removeToolbarIcons>
   <icon name="Register" />
   <icon name="Synchronize" />
   <icon name="SetSessionConfig" />
   <icon name="NewPwbWindow" />
   <icon name="DynamicActions" />
</removeToolbarIcons>
```

"queryDialogActions": optional

This tag includes the names of the buttons in the query dialog.

Example:

```
<queryDialogActions>
   <action name="OpenInPwbWindow" />
</queryDialogActions>
```

"sessionSettings": optional

The value contains the session settings of the PDM Workbench. The following entries are supported:

```
<queryMode name="listViewWindow" />
<queryMode name="pwbWindow" />
<relationDisplayMode name="relDisplay" />
<relationDisplayMode name="noRelDisplay" />
<passwordEncryption name="WinAuth" />
<passwordEncryption name="MD5" />
<passwordEncryption name="none" />
```

"installedLanguages": mandatory

This tag contains the name and date format of the installed languages (NLS files).

Example:

```
<installedLanguages visibleLength="15">
   <language name="en_us" displayName="NLS_EN"
             dateFormat="PWB_Standard" />
</installedLanguages>
```

"dateFormat": mandatory

This tag contains the definition of the date format for the installed languages.

Example:

```
<!-- PWB_Standard = YYYY-MM-DD -->
<dateFormat name="PWB_Standard" separator="-">
```

```
   <dateValue name="year" length="4" />
   <dateValue name="month" length="2" />
   <dateValue name="day" length="2" />
</dateFormat>
```

The order of the “dateValue” tags defines the format of the date. For each part of the date the length is defined in the tag.

"form" (Login Form): mandatory

This tag contains a description of the Login form. It defines the attributes needed for logging in to the PDM system. Generally it contains the attributes "login name", "password", and “database” at least, though other attributes like "group" can be defined if it is necessary for the PDM system.

Example:

```
<form name="Login" info="ShowOnlyLoginData">
   <frame displayName="NLS_UserData">
      <pwbFormAttribute name="PWBLoginUser" widgetType="ComboBox"
                        mode="update" visibleLength="10" required="true"
                        entryAllowed="true" dataSource="UserNames" />
      <pwbFormAttribute name="PWBLoginPassword"
                        widgetType="SingleLineEditor" mode="update"
                        visibleLength="10" required="false" />
      <formAttribute name="LoginDatabase" widgetType="ComboBox"
                     mode="update" visibleLength="10" required="true"
                     entryAllowed="false" />
   </frame>
</form>
```

The XML tags inside the "frame" tag describe how the attributes "user" and "password" are displayed in the login dialog.

"partClasses": optional

The value describes a list of class names (possibly of size 1) of all part classes. It has to be defined if the class can have sub parts.

Example:

```
<partClasses>
   <partClassName name="/Part/Assembly" canHaveSubParts="true"/>
   <partClassName name="/Part/Component" canHaveSubParts="false"/>
</partClasses>
```

"multiQuantityRelationAllowed": mandatory

The value indicates if the part-part relation created during the synchronize process will be created as multi quantity relation.

If there exists already a part-part relation between the two objects the same relation type will be used, not respecting the value.

Example:

```
<multiQuantityRelationAllowed value="true" />
```

For Aras Innovator the value has to be “true”.

"createRelationType": mandatory

The value indicates the type of the newly created part-part relation during the synchronize process.

Even if the multi quantity relation is allowed (see above) then a single quantity relation can be created.

Examples:

```
<createRelationType value="single"/>

<createRelationType value="multi"/>
```

For Aras Innovator the value has to be “multi”.

"standardPartFiles": optional

The attribute “download” defines whether the CATIA files which correspond to the standard parts should be downloaded to the local exchange map directory, and the attribute “localDir” defines the directory where the standard part CATIA files are searched if “download” is defined as “false”.

Example:

```
<standardPartFiles download="false" localDir="C:\StandardPartFiles" />
```

If a standard part administrator is creating new standard parts in PDM the setting “download” should be “true” so the files get uploaded correctly.

"standardPartUserDefPropAttribute": optional

The value defines the name of the user-defined CATIA attribute which indicates whether a CATPart represents a standard part. If the value of the attribute “name” is “true” or “1” then the CATPart is defined as a standard part. In that case the update process does not try to create a new part which corresponds to the CATIA document, but it queries for an existing standard part in PDM instead.

“catiaNodeBehaviorDefinitions ": optional

Defines whether embedded CATIA compoenents are skipped or ignored when a CATIA structure is processed.

Example:

```
<catiaNodeBehaviorDefinitions>

   <catiaNodeBehavior catiaNodeType="EmbeddedComponent"
                      partNumberPrefix="" behavior="SkipNode"/>

</catiaNodeBehaviorDefinitions>
```

“templateCatiaFilePath": optional

The path to the directory which contains the template CATIA files.

Example:

```
<templateCatiaFilePath value="C:\files\PDM Workbench\Templates" />
```

"object": 1 -n

This tag contains the definition of a PDM object class which can be used (queried, created, etc.) by the user.

The definition of PDM object classes, their corresponding dialogs and the actions that can be performed on them are described in the chapter **PDM Objects**.

"relation": 1 -n

This tag contains the definition of PDM relation classes that can be accessed (expanded, created, etc.) by the user.

The definition of PDM relation classes, their corresponding dialogs and the actions that can be performed on them are described in the chapter **PDM Relations**.

"attribute": 0 -n

The definition of PDM attributes that are referenced in dialogs.

Attributes and dialog forms are explained in the chapter **PDM Attributes and Form Attributes**.

"pvbAttribute": 0-n

The definition of attributes that do not correspond directly to PDM attributes of PDM objects.

Attributes and dialog forms are explained in the chapter **PDM Attributes and Form Attributes**.

"dataSource": 0-n

Data sources contain attribute values. By assigning data sources to attributes default values for these attributes can be defined.

Data sources are explained in the chapter **Data Sources**.

PDM Attributes and Form Attributes

Every PDM attribute that is displayed in a dialog form should be defined in an "attribute" tag.

The attribute definition contains the following attributes:

- "name" Mandatory, must correspond to the PDM attribute's name.
- "displayName" Mandatory. As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization.
- “dataSource” Optional. The data source includes the possible values for this attribute.
- "isFileName" Optional. If is set to "true" the value of the corresponding input filed is checked about illegal[2] characters when creating a file.
- "isPartNumber” Optional.
- “autoName” Optional.
- "isDerived" Optional.

Example:

```
<attribute name="name" displayName="NLS_Name" isFileName="true"
           isPartNumber="true" autoName="true"/>

<attribute name="current" displayName="NLS_current"
           dataSource="LifeCycleStates"/>

<attribute name="revision" displayName="NLS_Revision" />
```

[2] Filenames must not contain control characters, non printable characters and any of the following characters: `*?:;\/<>|`

A form definition contains form attributes which reference the previously defined PDM attribute.

The form attribute definitions contain the following attributes:

- "name" — Mandatory, must correspond to the PDM attribute's name.
- "displayName" — Optional. As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization. If not defined here the display name of the "attribute" tag will be used.
- "mode" — Possible values are "output" (read-only), "update" (can be modified), or "select" (e.g. for combo boxes).

  Default is "output".
- "visibleLength" — Optional, the length of the text editor widget in characters.
- "allowedLength" — Optional, the length of value that can be inserted in the text editor widget in characters.
- "required" — "true" or "false". If "true", then a value must be set.

  Default is "false".
- "widgetType" — Possible values are "SingleLineEditor", "MultiLineEditor", "ComboBox", "SingleCheckBox", "CheckBoxes", "RadioButtons", "SingleSelectorList", "MultiSelectorList", "NameValueList", "Date".

  Default is "SingleLineEditor".
- "embeddedObjAttr" — Optional. If the PDM attribute refers to a different PDM attribute in a contained object attribute, then this XML attribute's value contains that object attribute's name.
- "embeddedAttribute" — Optional. The name of the PDM attribute of the embedded object. If "embeddedObjAttr" is set, then "embeddedAttribute" must be set, too.
- "dataSource" — Optional. The value defines the link to a data source that is more special than the linked data source in the <attribute> tag.
- "listViewRelevant" — "true" or "false". If "true", then the attribute will appear in the query list view.

  This attribute should only be added to "Query" forms. Please refer to the user manual for more information.

  Default is "false".
- "sort" — Possible values are "true" or "false". If "true", then the attribute can be sorted in the query result window.
- "displayOnly" — Possible values are "true" or "false". If "true", then the display value of the value of the data source will be used.
- "entryAllowed" — Possible values are "true" or "false". If "true", then the user can enter a text additional to the attached data source.

Example:

```
<form name="Query">
```

```
<formAttribute name="name" widgetType="SingleLineEditor"
               mode="update" visibleLength="15" required="false"
               listViewRelevant="true" />
```

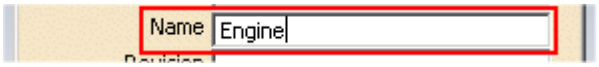
Form definitions generally refer to classes of PDM objects (query form, properties form, etc.). The definition of PDM object classes is described in chapter **PDM Objects**.

Description of the Widget Types

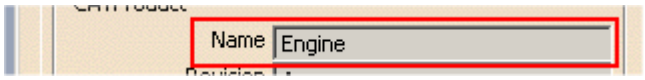
There are ten different widget types available to build up dialogs with. All widget types except “SingleLineEditor”, “MultiLineEditor” and one mode of “NameValueList” can only be used on attributes that have certain kinds of Data Sources attached. Data Sources are a container of a limited set of values.

The detailed explanation of Data Sources you can find in chapter **Data Sources**.

SingleLineEditor Supports "update" and "output" mode.
Can be used for attributes with no data source attached and also for attributes with data sources of type “SingleValue”.

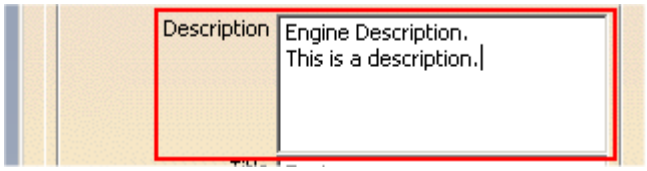


Picture 37: Single Line Editor Widget, update mode



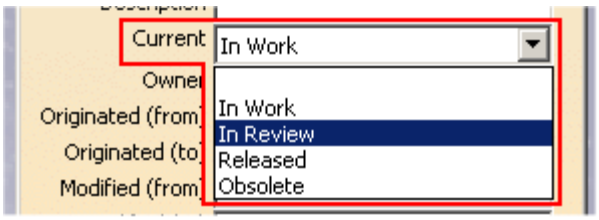
Picture 38: Single Line Editor Widget, output mode

MultiLineEditor Supports "update" and "output" mode.
Can be used for attributes with no data source attached and also for attributes with data sources of type “ValueList”.



Picture 39: Multi Line Editor Widget, update mode

ComboBox Supports “select” and “output” mode.
This widget type can only be used for attributes with data sources of type “ValueList”, “BooleanValueList” or “invokeMessage” if this message returns a set of values.



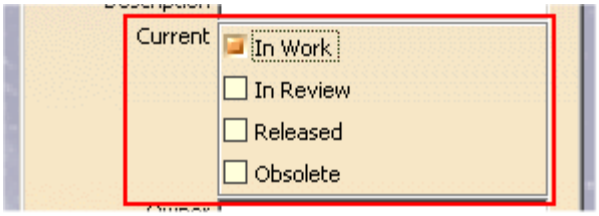
Picture 40: Combo Box Widget, select mode

SingleCheckBox Supports “select” and “output” mode.
Needs an attribute with a data source of type "BooleanValueList".
This widget should be used only for required attributes or for attributes that are only displayed, already set to a value and cannot be updated.

Picture 41: Single Check Box Widget, select mode

CheckBoxes

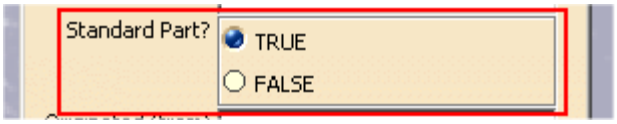
Supports “select” and “output” mode.
Possible for attributes with data sources which contain several values (type "ValueList”, “BooleanValueList” or “invokeMethod”) and where the user can select more than one value.



Picture 42: Check Boxes Widget, select mode

RadioButtons

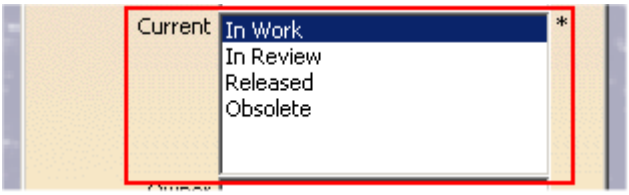
Supports “select” and “output” mode.
Possible for attributes with a data source of type "ValueList", “BooleanValueList” or “invokeMethod” where only one value can be selected at one time.
They should be used for required attributes only, because one value has always to be selected.



Picture 43: Radio Buttons Widget, select mode

SingleSelectorList

Supports “select” and “output” mode.
It represents a list with one column where one item is selectable.
It can be used for all attributes with attached data sources of type "ValueList", “BooleanValueList” or “invokeMethod”.



Picture 44: Single Selector List Widget, select mode

MultiSelectorList

Supports “select” and “output” mode.
It represents a list with one column where several items are selectable.
(The multi-selector list looks like the single-selector list, except that more than one item of the list can be selected.)
It can be used for all attributes with attached data sources of type "ValueList", “BooleanValueList” or “invokeMethod”.

NameValueList

Supports “select”, “update” and “output” mode.
It represents a list with two columns (e.g. for name value sets)

→ when working in update mode the widget type can be used for all attributes within the Schema file (no data source needed). The user can change every single column item (columns can be

empty).

→ when working in select mode the widget type can only be used for attributes with data sources of type “NameValueList” attached. (widget acts as a filter then)

PDM Objects

Object XML tags define PDM object classes that can be used in the PDM Workbench application. They represent the subset of objects defined within the PDM system which are needed in a PDM-CAD integration.

An "object" XML tag contains the following attributes:

- "name" The internal PDM class name.
- "displayName" The class name that is shown to the user.
- "icon" The icon that represents the class in the PDM window and the list window.

Example:

```
<object name="/Part/Assembly" displayName="NLS_Assembly" icon="Aras_Part">
```

For the icon to be displayed correctly in the PDM window a bitmap file with the name of the icon (in this example "Aras_Part.bmp") must exist in the subdirectory "resources\graphic\icons\normal" of the CATIA V5 directory (e.g. "intel_a" on Windows 32 Bit CATIA installation, and “win_b64” on Windows 64 Bit CATIA installation).

The list view window needs a bitmap file with the file name (icon name) + "16x16.bmp", in this example "Aras_Part16x16.bmp".

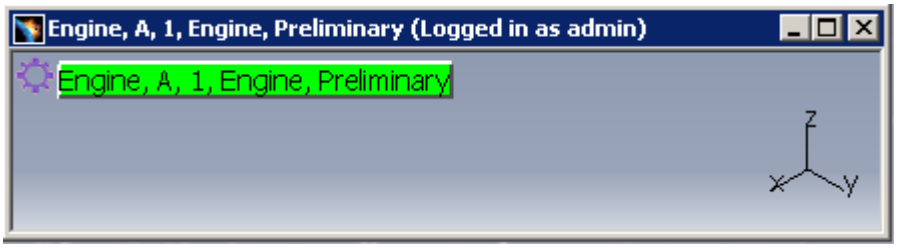
As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization.

Description of PDM Objects

The "description" tag defines which of the attributes of the class should be displayed beside the icon. In this example, these are the attributes “item_number”, “major_rev”, “generation”, “name”, and “state”.

Example:

```
<description>
   <descAttribute name="item_number" />
   <descAttribute name="major_rev" />
   <descAttribute name="generation" />
   <descAttribute name="name" />
   <descAttribute name="state" />
   <descAttribute name="classification" />
   <descAttribute name="description" />
</description>
```



Picture 45: PDM Node in PWB window

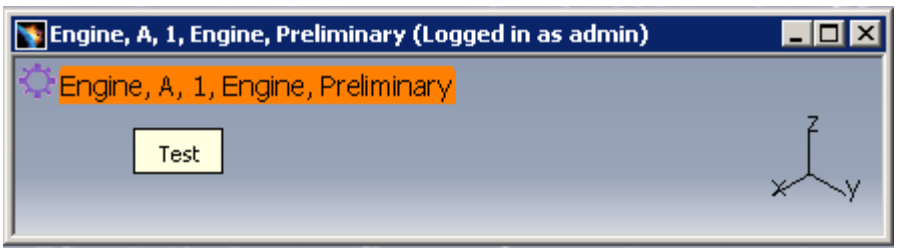
Tooltip of PDM Objects

The "tooltipAttribute" tag defines which string has to be shown as tooltip for the object. The value can be the name of an attribute of the object. If this value is empty then the description defined in the chapter above will be used.

Example:

```
<tooltipAttribute name="description" />
```

The attribute “description” has the value “Test”.



Picture 46: Tooltip of PDM Node in PWB window

Actions on PDM Objects

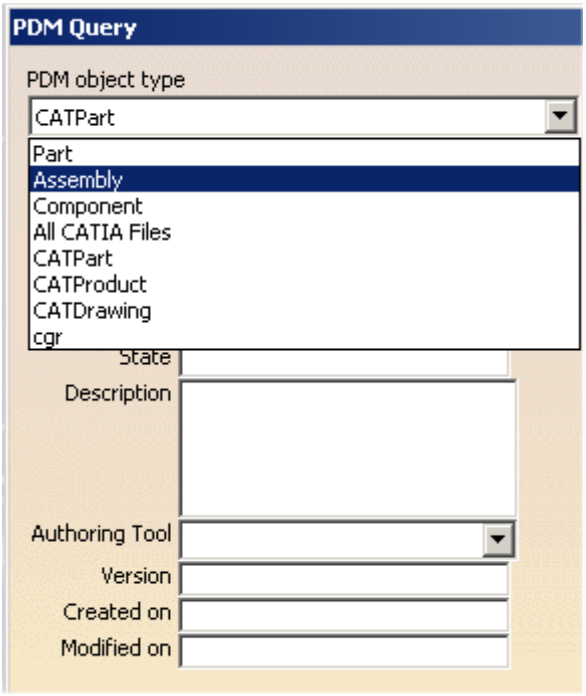
Actions that can be performed with PDM objects have to be defined in the Schema file. There are two kinds of actions: So-called toolbar actions, which are started by clicking on an icon in the PDM Workbench toolbar, and context actions, which are started by right-clicking on the node and selecting one of the context menu items.

Toolbar actions are defined with an "action" tag. The action "Query" can be defined on any object type.

Example:

```
<!-- * all PWB toolbar actions permitted for this object * -->

<action name="Query" />
```

If, for instance, the action "Query" is defined for the object type "/Part/Assembly", then, when the user clicks on the "Query" toolbar icon, the type "Assembly" (display name) is included in the query dialog list, otherwise it is not.



Picture 47: Select PDM object type in PDM Query dialog

Context Actions

Context actions are declared similarly to toolbar actions. For context actions, the tag "contextAction" is used.

Example:

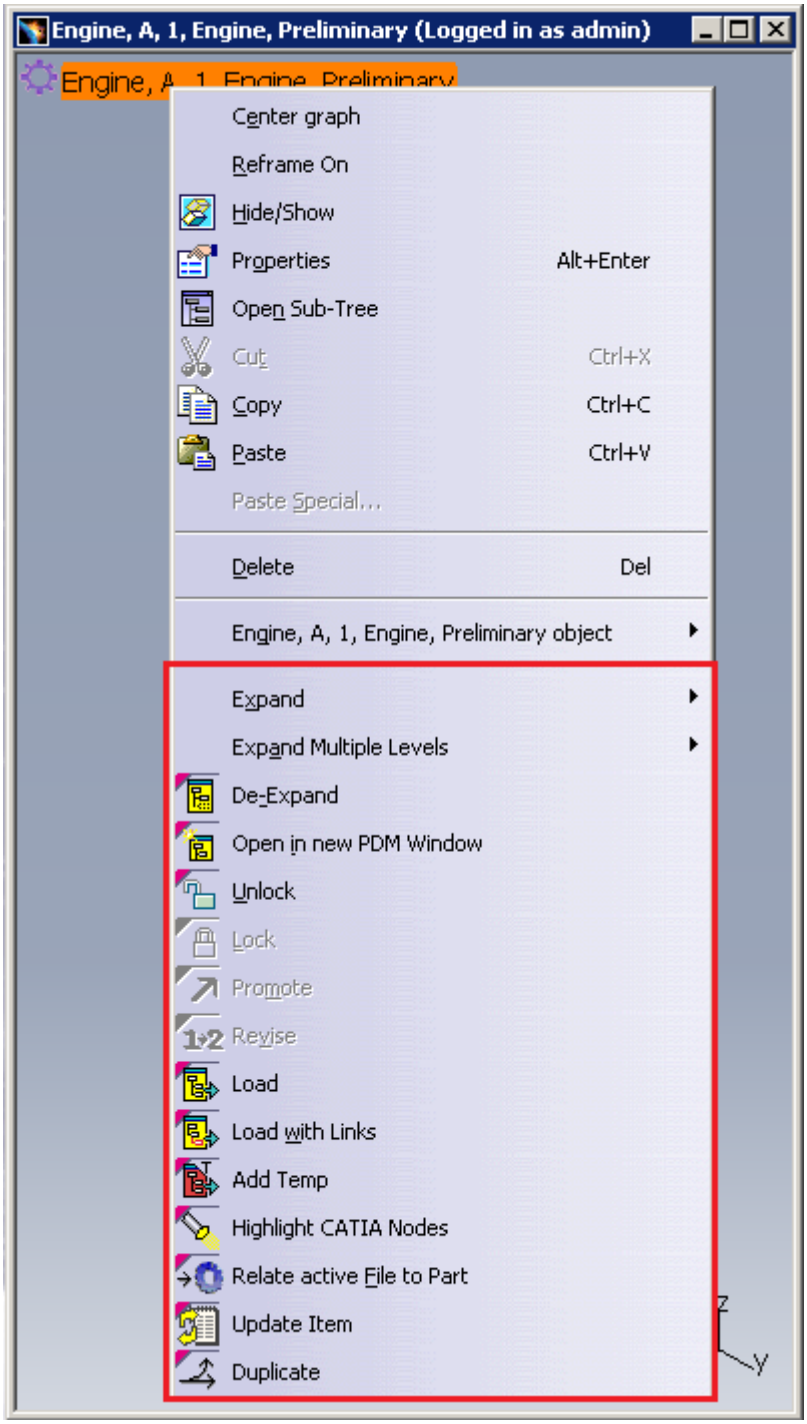
```
<!-- * all PWB context actions permitted for this object * -->
<contextAction name="Expand" usedIn="PdmWindow" />
<contextAction name="MultipleExpand" usedIn="PdmWindow" />
<contextAction name="DeExpand" usedIn="PdmWindow" />
<contextAction name="CheckIn" usedIn="PdmWindow|QueryDialog" />
<contextAction name="CheckOut" usedIn="PdmWindow|QueryDialog" />
<contextAction name="Promote" usedIn="PdmWindow|QueryDialog" />
<contextAction name="Revise" usedIn="PdmWindow|QueryDialog" />
<contextAction name="LoadStructure" usedIn="PdmWindow" />
<contextAction name="AddTemp" usedIn="PdmWindow" />
<contextAction name="Highlight" usedIn="PdmWindow" />
<contextAction name="UpdateSubCmpRels" usedIn="PdmWindow|QueryDialog" />
<contextAction name="UpdateParentCmpRel" usedIn="PdmWindow|QueryDialog" />
<contextAction name="UpdateItem" usedIn="PdmWindow" />
<contextAction name="DeleteItem" usedIn="PdmWindow" />
<contextAction name="DeleteNewestVerstion" usedIn="PdmWindow" />
```

The attribute "usedIn" defines where the context action is shown. The attribute value "PdmWindow" activates the action in the PDM window. The attribute value "QueryDialog" activates the action in the result list view of the query dialog. These two attribute values can be combined with the pipe "|".

The context action "Properties" has a window with two tabs for “Properties”, and "UpdateItem". The corresponding dialog forms must also be defined for that class.

The context action “Properties” is default and not explicitly defined in the Schema file.

If a certain context action is defined for a PDM object class in the Schema file, then the corresponding context menu entry for object nodes of that class exists.



Picture 48: Context actions for the type /Part/Assembly

Some things to keep in mind regarding the definition of context actions:

If the context action "Unlock" (CheckIn) is defined, then the dialog forms "CheckInNew" and "CheckIn" must also be defined.

If the context action "Lock" (CheckOut) is defined, then the dialog form "CheckOut" must also be defined.

The context action “Load” (LoadStructure) should only be defined on part objects.

Disabling context menu items in the CATIA structure window

In the corresponding “contextAction” definition the XML attribute “removeFromCatiaWindow” needs to be set to “true”:

```
<contextAction name="CheckIn" usedIn="PdmWindow|QueryDialog"
               removeFromCatiaWindow=”true” />
```

If that is done the corresponding context action will not be available in the CATIA window anymore. It may still be available as a context menu in the PDM structure window or as a context menu in the query dialog result list.

PDM Object Forms

The following forms can be defined for an object class:

"Query", "Properties", "UpdateItem", "CheckInNew", "CheckIn", "CheckOut"

The "CheckIn" and "CheckOut" forms do not have to contain any attributes, they just need to be defined.

Example:

```
<form name="Query">

   <formAttribute name="item_number" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="major_rev" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="generation" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="name" widgetType="SingleLineEditor" mode="update"
                  visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="state" widgetType="SingleLineEditor" mode="update"
                  visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="unit" widgetType="ComboBox" mode="update"
                  visibleLength="15" required="false" />

   <formAttribute name="make_buy" widgetType="ComboBox" mode="update"
                  visibleLength="15" required="false" />

   <formAttribute name="description" widgetType="MultiLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="created_on" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

   <formAttribute name="modified_on" widgetType="SingleLineEditor"
                  mode="update" visibleLength="15" required="false"
                  listViewRelevant="true" />

</form>
```

PDM Relations

Relation XML tags define PDM relation classes that can be used in the PDM Workbench application. They represent the subset of the relations defined in the PDM system that are needed in PDM-CAD integration.

As with the "object" XML tags, a "relation" XML tag contains the following attributes:

- "name" The internal PDM class name.
- "displayName" The class name that is shown to the user.
- "icon" The icon that represents the class in the PDM window and the list window.
- "createAllowed" Defines whether relations of this class can be created by the user. Relations are created by copying and pasting object nodes. If "createAllowed" is "true", then, at paste, the relation class is included in the list of applicable relations, otherwise it is not. Please refer to the user manual for more information.
- "expandAllowed" Defines if it is allowed to expand this relation class explicitly in the context action "Expand".

Example:

```
<relation name="Part BOM" displayName="NLS_PartBOM"
          icon="TcEnt_PartPartRelation" createAllowed="true"
          expandAllowed="true">
```

As for the PDM object classes, a bitmap file for the icon must exist in the CATIA icon subdirectory ("IconName.bmp" and "IconName16x16.bmp").

As described in "NLS Support for Display Names" the NLS string for the "displayName" XML attribute is defined in the CATNls file specific to the PDM system and the customization.

Description of PDM Relations

As with PDM object definitions, the "description" tag defines which of the attributes of the class should be displayed beside the icon.

In this example the attributes "Class", "major_rev", and "generation" are defined as the description.

Example:

```
<description>
   <descAttribute name="Class" />
   <descAttribute name="major_rev" />
   <descAttribute name="generation" />
</description>
```

"Relationship" tags

The tags "leftToRightRelationship" and "rightToLeftRelationship" define the relationships for the two sides of the relation. They contain the following XML attributes:

- "name" The internal PDM relationship name.
- "displayName" The relationship name that is shown to the user.
- "multipleExpand" If "multipleExpand" is "true", then the relationship appears in the "Expand Multiple Levels" context menu, otherwise it does not. For this the context action "MultipleExpand" must be defined on the PDM object. The default value is "false".

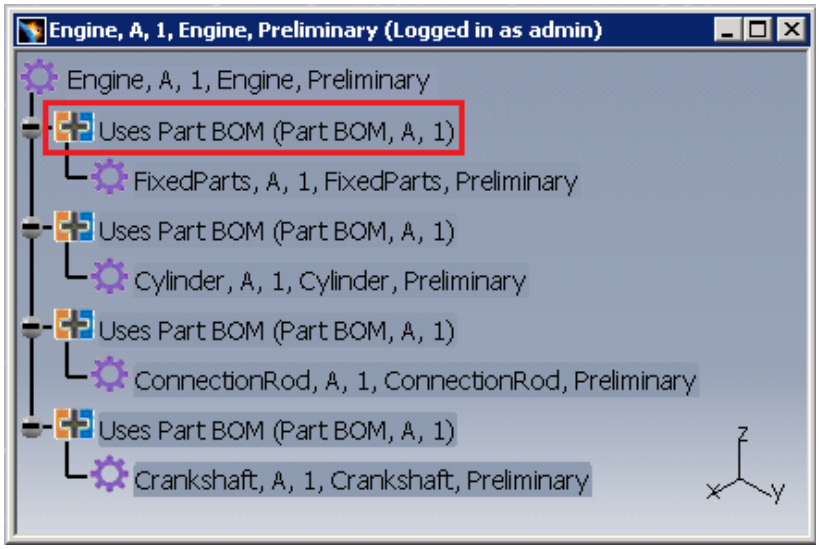
Example:

The Aras relation "Part BOM" has the relationships "Part BOM_LeftToRight" (display name is "Uses Part BOM") and "Part BOM_RightToLeft" (display name is "Is Used in Part BOM").

```
<leftToRightRelationship name="Part BOM_LeftToRight"
                         displayName="NLS_PartBOM_LeftToRight"
                         multipleExpand="true" />
```

```
<rightToLeftRelationship name="Part BOM_RightToLeft"
                         displayName="NLS_PartBOM_RightToLeft"
                         multipleExpand="true" />
```

The text that describes a relation icon is the display name of the expanded relationship ("Part BOM“ in this example) as well as the parameters defined in the "description" XML tag in parentheses “Class”, “major_rev”, and “generation” in this example).



Picture 49: Relation icon with relationship and description attribute

“Left and Right Object” Classes

The tags "leftObject" and "rightObject" define which object classes are valid for this relation.

In the following example the relation is between CATIA parts:

```
<leftObject name="/Part/Assembly" />
<rightObject name="/Part/Assembly" />
<rightObject name="/Part/Component" />
```

PDM Relation Forms

The following forms can be defined for a relation class:

"Properties", "UpdateItem", "Create"

The definition of the dialog forms for relations is similar to the definition of the dialog forms for objects.

Example:

```
<form name="UpdateItem">
   <formAttribute name="reference_designator"
                  widgetType="SingleLineEditor" mode="update"
                  visibleLength="15" />
</form>
```

Data Sources

Data sources describe a static set of values that are already known when writing the Schema file. The set of these values will never change during the lifetime of the PDM Workbench.

Data Source "Value" tag

The value tag of static data sources contains the following XML tags:

- "name" The PDM name of the attribute.
- "displayName" The dialog display name of the attribute.
- "booleanValue" "true" or "false" to assign the correct value to the attribute names (this tag is only used for type BooleanValueList).
- "valueName" The PDM name of the value attribute (this tag is only used for type NameValueList).
- "displayValue" The dialog display name of the value attribute (this tag is only used for type NameValueList).

Static data sources can be of type:

SingleValue: the data source contains only one static element.

Example:

```
<dataSource name="Autoname" type="SingleValue">
   <value name="autoname" displayName="NLS_Autoname" />
</dataSource>
```

ValueList: the data source contains a set of static value elements.

Example:

```
<dataSource name="LifeCycleStates" type="ValueList">
   <value name="Preliminary" displayName="NLS_Preliminary"/>
   <value name="Review" displayName="NLS_Review"/>
   <value name="Approve" displayName="NLS_Approve"/>
   <value name="Released" displayName="NLS_Released"/>
</dataSource>
```

BooleanValueList: the data source contains exactly the value pair "true" and "false".

Example:

```
<dataSource name="PlusOrMinus" type="BooleanValueList">
   <value name="+" displayName="NLS_ValueSetPlus" booleanValue="true" />
   <value name="-" displayName="NLS_ValueSetMinus" booleanValue="false" />
</dataSource>
```

NameValueList: the data source contains a list of name-value pairs.

Example:

```
<dataSource name="Test" type="NameValueList">
   <value name="test1" displayName="NLS_test1" valueName="test1_value"
          displayValue="NLS_test1_value" />
   <value name="test2" displayName="NLS_test2" valueName="test2_value"
          displayValue="NLS_test2_value" />
</dataSource>
```

Complete example of using a datasource tag:

The attribute “CheckedOut” can be assigned exactly to “true” or “false”. Within the dialog this will be expressed by showing a “+” or a “-“ sign. Therefore we define a data source called “PlusOrMinus” and attach this container to the attribute description.

```
<attribute name="CheckedOut" displayName="NLS_CheckedOut"
           dataSource="PlusOrMinus" />

<dataSource name="PlusOrMinus" type="BooleanValueList">

   <value name="+" displayName="NLS_ValueSetPlus" booleanValue="true" />

   <value name="-" displayName="NLS_ValueSetMinus" booleanValue="false" />

</dataSource>
```

Configurations for specific functionalities

Basic Drawing Link Support

The drawing link functionality has to be switched on by defining the schema file setting

```
<updateCatiaLinksInPdm value="true" />
```

In addition to that, an additional form attribute has to be defined in the “Register” form of the CATDrawing object definition. This attribute has to have the attribute displayOnly=”true” to indicate that it is not an actual PDM attribute, only a display attribute in the dialog:

```
<formAttribute name="LinkedMainGeometryFile" widgetType="ComboBox"
               mode="update" visibleLength="15" required="false"
               listViewRelevant="false" displayOnly="true" />
```

The corresponding attribute definition has to contain the XML attribute pwbAttrInfo="MainDrwGeometryFile", and it has to refer to a data source which contains some special definitions:

```
<attribute name="LinkedMainGeometryFile"
           displayName="NLS_LinkedMainGeometryFile"
           dataSource="LinkedGeometryFiles"
           pwbAttrInfo="MainDrwGeometryFile" />
```

The data source definition which is referred has to contain the definition additionalValues="GeometryFilesLinkedInSession":

```
<dataSource name="LinkedGeometryFiles" type="ValueList"
            additionalValues="GeometryFilesLinkedInSession" />
```

If the configuration is set up correctly then the user will see an additional list widget in the register dialog for CATDrawings which enables him to select one of the linked 3D geometry files (see User_Manual_Aras).

Basic Multi-Model Link Support

Like the drawing link functionality, the multi-model link functionality has to be switched on by defining the schema file setting

```
<updateCatiaLinksInPdm value="true" />
```

Management of CATIA templates in Innovator

Items in Aras Innovator:

CAD documents with the attribute ‘is_template’ having the value ‘1’ and their related file items have to be imported to PDM. This can be done with a batch process, or with the PDM Workbench integration itself.

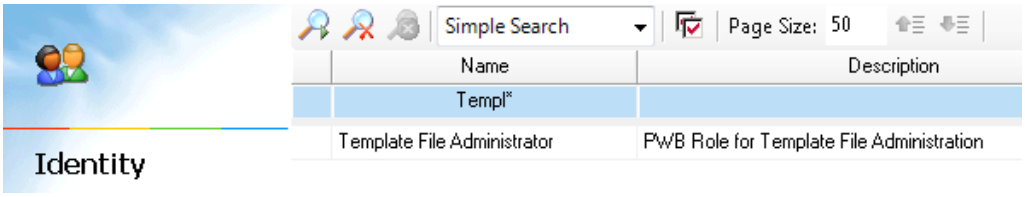
For this, the user has to be in a special template file administrator group.

This group can be configured as follows:

The configuration variable ‘TemplateFileAdmin’ in the ‘Settings’ tab of the active PWB configuration item has to contain the name of an identity as its value, for instance ‘Template File Administrator’:

TemplateFileAdmin	Template File Administrator

Picture 50: Template file administrator configuration variable



Picture 51: Example template file administrator identity

If this identity contains the user which is currently logged in then the user is a template file administrator and has the rights to create and modify template file documents.

For regular users existing template file documents are read-only (black text background).

One way of creating template file documents in Innovator with PDM Workbench is this:

1. Make sure the following settings are set in the active PWB configuration item, at least temporarily:

UseBomPartStructure	false
ShowCreateDialogsDuringUpdate	true

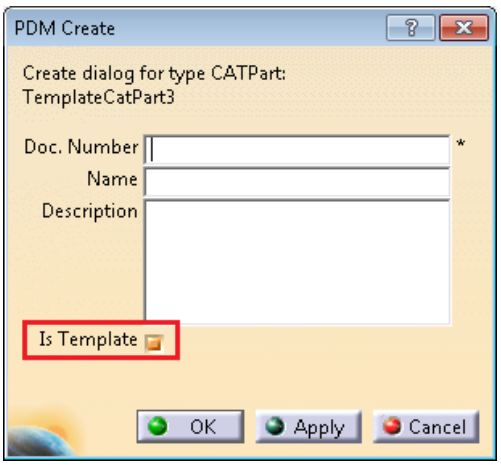
Picture 52: PWB Configuration variables

2. Log in to Innovator with PDM Workbench as a template file administrator.

3. Open a CATIA file which is supposed to be used as a PDM template document in CATIA V5.

4. Click on the Update icon in the PDM Workbench toolbar to create the CAD document and to upload the file.

Since the two setting are set to the values described in step 1 only a CAD document will be created, and a create dialog is shown to the user:

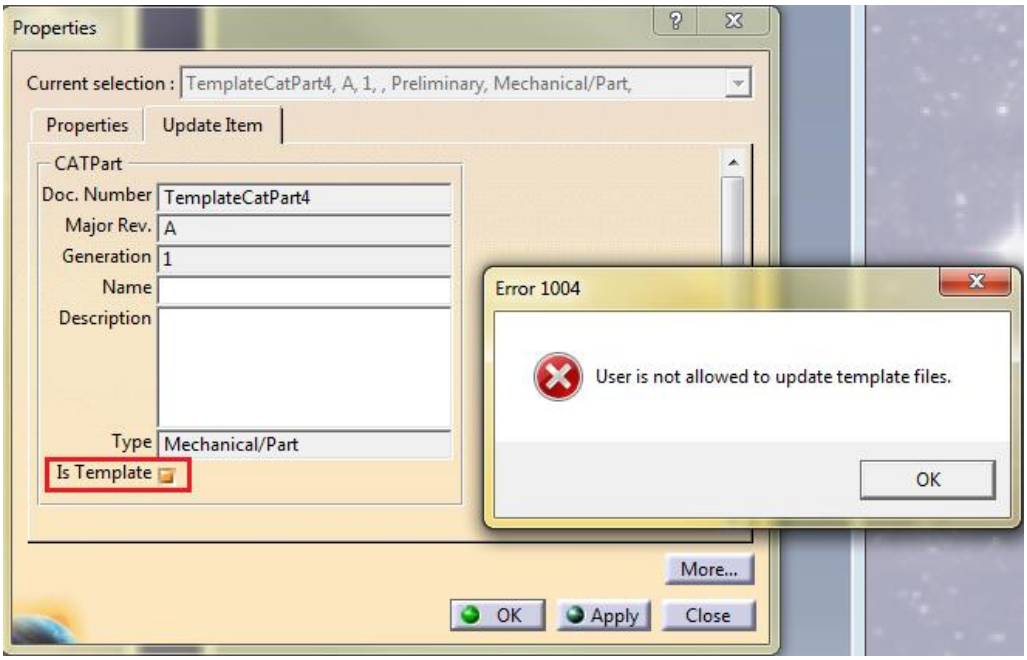


Picture 53: Create dialog containing “Is Template” checkbox

The document number has to be entered, and it is important that the 'Is Template' attribute is checked.

Another possibility would be to first create the template file as a regular file, and to change the 'is_template' attribute afterwards. In this case the 'ShowCreateDialogsDuringUpdate' setting does not have to be set to 'true'. Even the 'UseBomPartStructure' setting can be set to 'true', but in this case a Part item is created too. This Part item may have to be deleted later, because it should probably not be in the bill of materials.

Only a template file administrator can change the 'is_template' attribute value of a CAD document item. A regular user will get this error message:



Picture 54: Template file creation error message

For a regular user template file CAD document items are read-only and they also can not be locked by a regular user.

PWBSchema.xml configuration:

In the schema file the configuration 'templateFiles' has to exist and it has to contain the attribute ' loadFrom="PDM" ':

```
<templateFiles loadFrom="PDM"
               catPrt="PwbCatPrtTmplFileNames"
               catPrd="PwbCatPrdTmplFileNames"
               catDrw="PwbCatDrwTmplFileNames" />
```

The values of the XML attibutes 'catPrt', 'catPrd', and 'catDrw' have to contain the names of data source definitions of the type 'ValueList' which contain the file names of the CATPart, CATProduct, and CATDrawing template files:

```
<dataSource name="PwbCatPrtTmplFileNames" type="ValueList">
      <value name="TemplateCatPart1.CATPart" />
      <value name="TemplateCatPart2.CATPart" />
</dataSource>

<dataSource name="PwbCatPrdTmplFileNames" type="ValueList">
      <value name="TemplateCatProduct1.CATProduct" />
      <value name="TemplateCatProduct2.CATProduct" />
```

```
</dataSource>


<dataSource name="PwbCatDrwTmplFileNames" type="ValueList">
      <value name="TemplateCatDrawing1.CATDrawing" />
      <value name="TemplateCatDrawing2.CATDrawing" />
</dataSource>
```

The first file in each of these lists is the default template, and that template file is used when the template file functionality is switched on, and if a part structure with no related CAD documents is loaded to CATIA.

The old template file functionality, where locally accessible files are used, is turned on by a definition like this in the schema file:

```
<templateFiles loadFrom="C:\Users\Public\PDM-Workbench\Templates"
               catPrt="PwbCatPrtTmplFileNames"
               catPrd="PwbCatPrdTmplFileNames"
               catDrw="PwbCatDrwTmplFileNames" />
```

In this definition the absolute file path which is defined as the value of the 'loadFrom' XML attribute has to contain the CATIA files which are defined in the data sources which are referenced in the XML attributes 'catPrt', 'catPrd', and 'catDrw'.

If 'loadFrom="PDM"' is set, then CAD documents with the same document number (without the file extension) and the PDM attribute 'is_template' having the value '1' have to be defined in Innovator, and they have to be related to File items with the same file name (with the file extension).

Additionally the 'Create' dialog definitions for the PDM types for CATPart, CATProduct, and CATDrawing files need to contain

Here an example is given for the CATPart type:

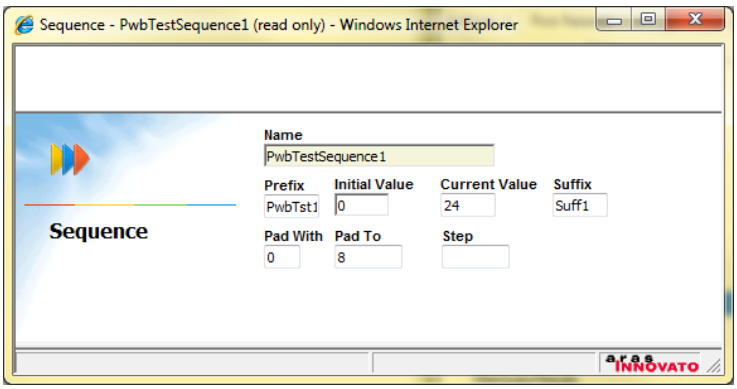
```
<form name="Create">
      <formAttribute name="CatPrtTemplate" widgetType="ComboBox"
                     mode="update" visibleLength="30" required="false" />
</form>
```

The referenced attribute with the name "CatPrtTemplate" needs to contain two specific definitions: A data source which contains the names of the template files (as described above), and a specific information that this attribute contains information about CATPart template files (shown in red below).

```
<attribute name="CatPrtTemplate" displayName="NLS_CatPrtTemplate"
           dataSource="PwbCatPrtTmplFileNames"
           pwbAttrInfo="CatPrtTemplateListAttr"/>
```

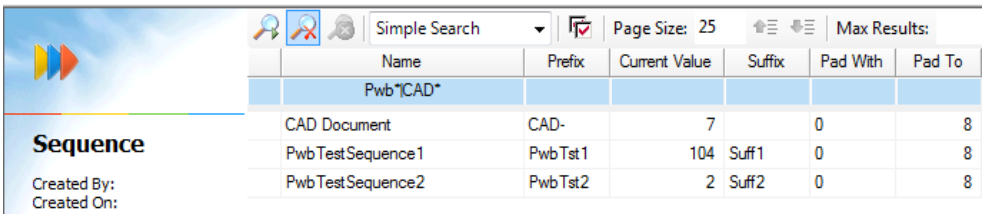
Autoname Support using Innovator Sequence Items

First sequence Innovator items which should be used for the autoname functionality need to be created:



Picture 55: Sample sequence item

The following sequence items will be used in the configuration example:



Name	Prefix	Current Value	Suffix	Pad With	Pad To
Pwb*|CAD*					
CAD Document	CAD-	7		0	8
PwbTestSequence1	PwbTst1	104	Suff1	0	8
PwbTestSequence2	PwbTst2	2	Suff2	0	8

Picture 56: Sequence items used in example

Then the sequence items to be used need to be configured in the schema file:

First an attribute with a data source which contains the names of the sequence items needs to be defined:

```
<attribute name="pwbAutonameRule" displayName="NLS_AutonameRule"
           dataSource="AutonameRules" />


<dataSource name="AutonameRules" type="ValueList">
       <value name="PwbTestSequence1" displayName="" />
       <value name="PwbTestSequence2" displayName="" />
       <value name="CAD Document" displayName="" />
</dataSource>
```

Then a corresponding form attribute has to be included in the login dialog …

```
<form name="Login" info="ShowOnlyLoginData" >
       <frame displayName="NLS_UserData">
              ...
              <formAttribute name="pwbAutonameRule"
                             widgetType="ComboBox"
                             mode="update" visibleLength="15"
                             required="false" entryAllowed="false" />
       </frame>
</form>
```

and in the “Set PDM Configuration” dialog.

```
<form name="PdmSessionConfig">
       <formAttribute name="pwbAutonameRule" widgetType="ComboBox"
                      mode="update" visibleLength="15" required="false"
                      listViewRelevant="true" />
</form>
```

This will enable the user to select a sequence item name as an autoname rule either at login or later while working in the PDM Workbench session.

In order for the “Set PDM Configuration” dialog to appear the setting "SetSessionConfig" has to be removed from the “removeToolbarIcons” definition:

```
<removeToolbarIcons>
      <!-- "Register", "Update", "Synchronize", "Refresh",
           "SetSessionConfig", "NewPwbWindow", "DynamicActions", "About"
        -->
      <icon name="Register" />
      <icon name="Synchronize" />
      <icon name="NewPwbWindow" />
      <!-- <icon name="SetSessionConfig" /> -->
      <icon name="DynamicActions" />
</removeToolbarIcons>
```

CATDrawing attribute mapping

The drawing attribute mapping functionality is configured by defining the configuration attributes “CatiaDrawingCadFileAttributes”, which defines a list of CATDrawing attributes, and “CatiaDrawingPdmDocAttributes”, which defines the corresponding list of PDM attributes on the CATDrawing item in Innovator. The value strings can contain any number of attribute names, divided by the “|” character.

This is a sample configuration:

CatiaDrawingCadFileAttributes	DrwAttr1|DrwAttr2|DrwAttr3
CatiaDrawingPdmDocAttributes	item_number|name|major_rev

Picture 57: Sample CATDrawing attribute mapping configuration

Configurable CATIA components support

It is possible to load and update CATProduct structures which contain embedded CATIA components. Depending on the part number prefix the embedded component nodes can either be “skipped”, that is, the node is treated as if it did not exist, but its child nodes are processed, or the can be “ignored”, that is, the node and all its child nodes are treated as if they did not exist.

The configurable node behavior functionality has to be switched on by defining the schema file setting “catiaNodeBehaviorDefinitions”:

```
<catiaNodeBehaviorDefinitions>
      <catiaNodeBehavior catiaNodeType="EmbeddedComponent"
                         partNumberPrefix="SKIP_" behavior="SkipNode"/>
      <catiaNodeBehavior catiaNodeType="EmbeddedComponent"
                         partNumberPrefix="IGN_" behavior="IgnoreNode"/>
</catiaNodeBehaviorDefinitions>
```

The behavior settings “SkipNode” and “IgnoreNode” are allowed.

Default sort criteria for query results

For every PDM item type an attribute can be configured to be the default sort order attribute. Query results for that item type are automatically sorted by that attribute.

The ‘object’ definiction can contain a new optional XML attribute: “defaultSortAttribute”:

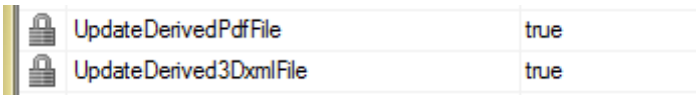
```
<object name="Part" displayName="NLS_Part"
        icon="Aras_Part" defaultSortAttribute="item_number" >
```

This attribute has to contain the internal name (not the CATNls name) of an attribute of the item type, e.g. ‘item_number’. The query result will be sorted by the values of that attribute.

Derived viewable files

PDF files derived from CATDrawings, and 3DXML files derived from CATParts can optionally be generated and uploaded at every Update.

In the active PWB Configuration item the settings ‘UpdateDerivedPdfFile’ and ‘UpdateDerived3DxmlFile’ have to be set to ‘true’:



Picture 58: Sample Derived Viewable Files configuration

Support for relating a new CATIA file to an existing Part

The currently active CATIA document (only CATParts or CATDrawings) can be related to an existing BOM part item. If there is already a corresponding CAD document related to the part the document’s file can be overwritten.

For part items the context action ‘RelateToPart’ can be added:

```
<contextAction name="RelateToPart" usedIn="QueryDialog"/>
```

“Delete relation” context menu action in the PDM structure window

PDM relations can be deleted in the PDM structure window with a single context menu action now, even if the PDM relations are not displayed in the structure.

For any item the context action ‘DeleteRelation’ can be added:

```
<contextAction name="DeleteRelation" usedIn="PdmWindow" />
```

The Autoname functionality can use a server method

The Autoname functionality can use a server method instead of using a sequence item directly for obtaining a PDM-generated part or document number value.

The autoname functionality has to be configured in the PWBSchema xml file.

The setting ‘UseServerMethodsForAutoname’ has to be set to ‘true’ in the active PWB Configuration item.

Additionally server methods whose names correspond to the names configured in the PWBSchema.xml file has to be defined on the Innovator server.

The server method can use information from standard CATIA attributes of the CATIA files to be imported to PDM, or values from PDM Workbench dialogs.

This is an example of such a server method:

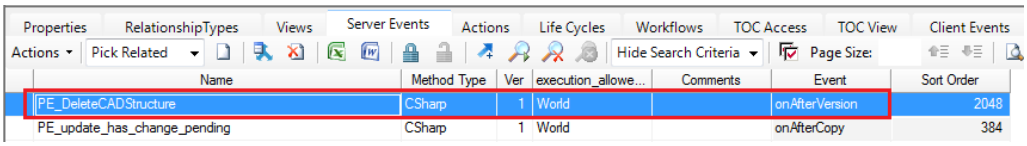
```
ArasUtil ArasUtilObj = new ArasUtil();
Innovator InnovatorObj = this.getInnovator();

// Preparing the input information
```

```
string Autoname = this.getProperty("Autoname");
string PdmType = this.getProperty("Type");
string PdmClassification = this.getProperty("Classification");

Item CatiaStdPropsItem = this.getPropertyItem("CatiaStdProps");
IDictionary<string, string> CatiaStdPropsDict = null;
if (CatiaStdPropsItem != null)
{
    CatiaStdPropsDict =
       ArasUtilObj.DialogAttrsItemToDictionary(CatiaStdPropsItem);
}

Item CadDocInputDialogItem = this.getPropertyItem("CadDocDialogAttrs");
IDictionary<string, string> CadDocInputDialogDict = null;
if (CadDocInputDialogItem != null)
{
    CadDocInputDialogDict =
       ArasUtilObj.DialogAttrsItemToDictionary(CadDocInputDialogItem);
}

Item PartInputDialogItem = this.getPropertyItem("PartDialogAttrs");
IDictionary<string, string> PartInputDialogDict = null;
if (PartInputDialogItem != null)
{
    PartInputDialogDict =
       ArasUtilObj.DialogAttrsItemToDictionary(PartInputDialogItem);
}

// Place the custom algorithm here
string OutputLogInfo = "";

OutputLogInfo += "Autoname:'" + Autoname + "'" + "|";
OutputLogInfo += "PdmType:'" + PdmType + "'" + "|";
OutputLogInfo += "PdmClassification:'" + PdmClassification + "'" + "|";

if (CatiaStdPropsDict != null)
{
    OutputLogInfo += "CatiaStdProps:" + "|";
    var Enumerator = CatiaStdPropsDict.GetEnumerator();
    while (Enumerator.MoveNext() == true)
    {
        var CurrentVal = Enumerator.Current;
        OutputLogInfo += "'" + CurrentVal.Key + "'->'" +
               CurrentVal.Value + "'" + "|";
    }
}

if (CadDocInputDialogDict != null)
{
       OutputLogInfo += "CadDocInputDialog:" + "|";
    var Enumerator = CadDocInputDialogDict.GetEnumerator();
    while (Enumerator.MoveNext() == true)
    {
        var CurrentVal = Enumerator.Current;
        OutputLogInfo += "'" + CurrentVal.Key + "'->'" +
               CurrentVal.Value + "'" + "|";
    }
}

if (PartInputDialogDict != null)
{
       OutputLogInfo += "PartInputDialog:" + "|";
    var Enumerator = PartInputDialogDict.GetEnumerator();
    while (Enumerator.MoveNext() == true)
    {
        var CurrentVal = Enumerator.Current;
        OutputLogInfo += "'" + CurrentVal.Key + "'->'" +
               CurrentVal.Value + "'" + "|";
    }
}

// Getting the actual autoname value
string AutonameValue =
       ArasUtilObj.GetNextAutonameSequence(InnovatorObj, Autoname);

OutputLogInfo += "done";

// Preparing the output
```

```
IDictionary<string, string> OutputInfoDict = new Dictionary<string, string>();

OutputInfoDict.Add("AutonameValue", AutonameValue);

OutputInfoDict.Add("LogLines", OutputLogInfo);

Item OutputInfoItem = ArasUtilObj.DialogAttrsDictionaryToItem(InnovatorObj,
OutputInfoDict);

return OutputInfoItem;
```

The PE_DeleteCADStructure server event method can be removed

The PE_DeleteCADStructure server event method, which is used for deleting the CAD Structure relations of a new generation of a CAD Document item, can be removed.

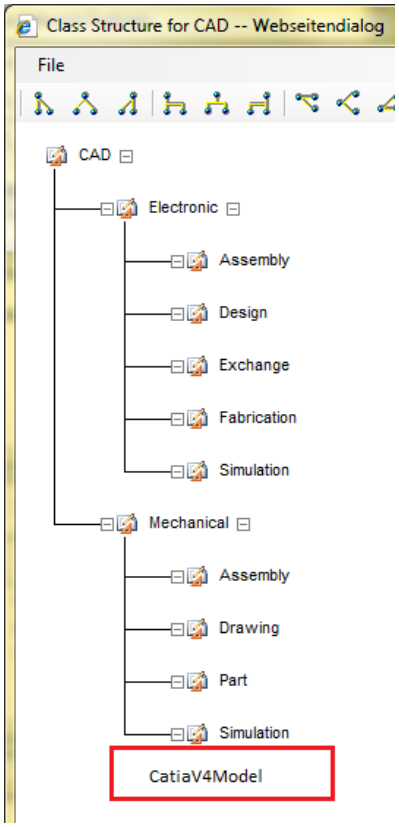


Picture 59: The PE_DeleteCADStructure server event method

If the server event method 'PE_DeleteCADStructure' is removed then the setting 'CadStructRelsDeletedAfterVersion' has to be set to 'false' in the active PWB Configuration item.

Using CATIA V4 models

If structures which contain CATIA V4 models are used a new classification for the item type CAD needs to be defined, for instance 'Mechanical/CatiaV4Model'.



Picture 60: Class structure for type 'CAD'

This classification has to be the same as the one defined in the PWBSchema.xml client configuration file. e.g.:

```
<object name="/CAD/Mechanical/CatiaV4Model" displayName="NLS_model" icon="model">
```

Support for the new CAD structure instance handling introduced in Innovator 9.4 and 10.0

A new relation with the name “CAD Instance” has been introduced, which contains instance information for “CAD Structure” relations.

By default the new CAD Instance relation is used to store CATIA instance information.

If the setting ‘UseCadInstance’ in the used PWB Configuration item is set to ‘false’ then the old instance handling (using three attributes on ‘CAD Structure’ to store the instance name, the instance description, and the transformation matrix) is used.

Standard Part Functionality

In part structure mode, it is possible to define part items and their corresponding CAD document items as standard parts. Standard parts are supposed to be parts which are used in a wide variety of different contexts and which are generally not modified by the designer, only used in the product structures that the designer works on.

Only a standard part administrator can create or modify standard parts. A standard part administrator is defined as every identity which belongs to the identity which is set as the setting “StandardPartAdmin”. By default it is the identity “Standard Part Administrator”, which is added with the PDM Workbench installation:

StandardPartAdmin	Standard Part Administrator

Picture 61: Sample Standard Part Administrator configuration

By default the “Innovator Admin” identity is a standard part administrator.

The CATParts which are related to standard parts can either be downloaded like the CATIA files which are related to other parts, or they can reside in a local directory which is accessible to CATIA V5.

This setting in the PWBSchema.xml file defines that the standard part CATParts are downloaded to the PWB_XMAP directory:

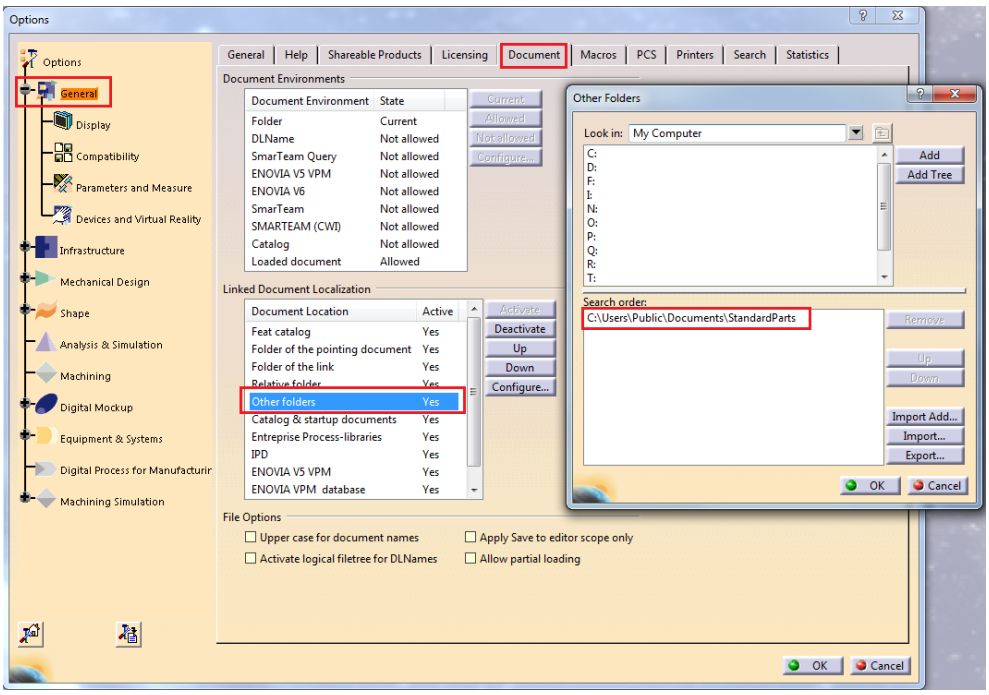
```
<standardPartFiles download="true" />
```

This setting defines that they are not downladed, but taken from a local directory:

```
<standardPartFiles download="false"
                   localDir="C:\Users\Public\Documents\StandardParts" />
```

For this setting three conditions need to be fulfilled:

1. The defined directory is accessible in the CATIA V5 environment.
2. The standard part CATPart files reside in that directory.
3. CATIA V5 needs to be set up to be able to use files from that directory.



Picture 62: Making CATParts in a local folder accessible

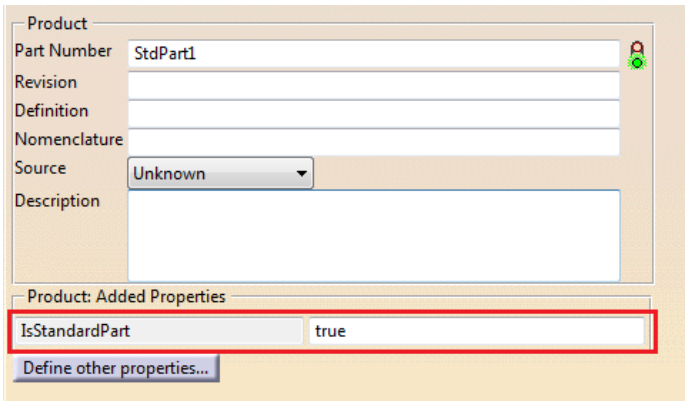
Another setting makes it possible to import structures containing standard part CATParts to PDM even if those standard parts have not been loaded in the PDM Workbench session.

Normally CATIA files which are not loaded from PDM are treated as new items, and new CAD documents and parts are created for them in the update process. If CATParts contain a certain definition as a user-defined attribute they are treated as standard parts, and the update process queries for existing standard part items in the database instead of trying to create a new part item. If such a standard part item is found it is related to the parent part.

In the PWBSchema.xml file it is possible to define the name of the user-defined attribute which defines a standard part CATPart file, for instance 'IsStandardPart':

```
<standardPartUserDefPropAttribute name ="IsStandardPart" />
```

A user-defined attribute with the same name has to be defined in the CATPart file. If the value is any of the strings "true", "TRUE", or "1", then the CATPart is considered to be a standard part CATPart.



Picture 63: Defining a CATPart as a standard part

Check for CAD document CATIA release at PDM update

A new functionality optionally asks the user before overwriting a file which has been created with a lower release of CATIA V5.

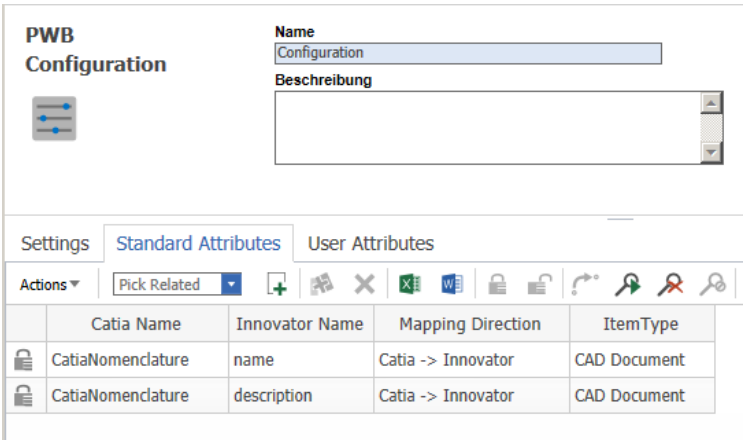
The functionality can be switched on with this setting in the PWBSchema.xml file:

```
<checkAuthoringToolVersion value="true"/>
```

Extended attribute mapping functionality

It is possible to define multiple assignments for the same CATIA property in the CAD to PDM direction.

The attribute mapping in the PWB Configuration item can contain multiple PDM attributes for the same CATIA attribute.



Picture 64: Example PWB Configuration attribute mapping

LocalWorkspace Information

It is possible to check the status of the CATIA documents which are downloaded to the local working directory (PWB_XMAP). A list displays the local files and information about their corresponding CAD documents in PDM if they exist.

The icon which opens the window which contains the list of files/documents can be switched on by removing the entry “LocalWorkspace” from the ‘removeToolbarIcons’ definition in the PWBSchema.xml file:

```
<removeToolbarIcons>
      <!-- <icon name="LocalWorkspace" /> -->
      <icon name="Register" />
      <icon name="Synchronize" />
      <icon name="NewPwbWindow" />
      <icon name="DynamicActions" />
</removeToolbarIcons>
```

The PDM attributes which are shown can be configured in the PWBSchema.xml file.

This is an example configuration:

```
<form name="LocalWorkspace" defaultSortAttribute="filename">
      <formAttribute name="modified" visibleLength="8" required="false"
                     listViewRelevant="true" />
      <formAttribute name="filename" visibleLength="10" required="false"
                     listViewRelevant="true" />
      <formAttribute name="item_number" visibleLength="10"
                     required="false" listViewRelevant="true" />
```

```
        <formAttribute name="major_rev" visibleLength="6" required="false"
                       listViewRelevant="true" />

        <formAttribute name="generation" visibleLength="6" required="false"
                       listViewRelevant="true" />

        <formAttribute name="name" visibleLength="10" required="false"
                       listViewRelevant="true" />

        <formAttribute name="state" visibleLength="10" required="false"
                       listViewRelevant="true" />

        <formAttribute name="description" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="created_on" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="modified_on" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="created_by_id" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="modified_by_id" visibleLength="10"
                       required="false" listViewRelevant="true" />

        <formAttribute name="locked_by_id" visibleLength="10"
                       required="false" listViewRelevant="true" />

</form>
```